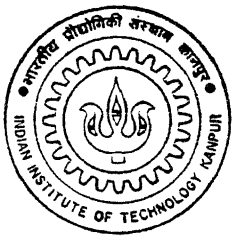
MULTIMODAL AND MULTIOBJECTIVE OPTIMIZATION USING REAL-CODED GENETIC ALGORITHMS

by

Amarendra Kumar



DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

FEBRUARY, 1996

MULTIMODAL AND MULTIOBJECTIVE OPTIMIZATION USING REAL-CODED GENETIC ALGORITHMS

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by

Amarendra Kumar

to the

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

February, 1996

Dedicated
to
my parents

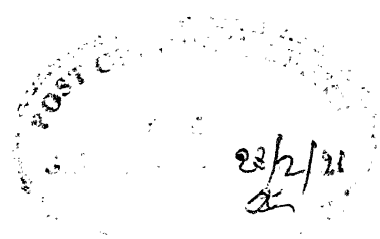
CERTIFICATE

It is certified that the work contained in the thesis entitled "**Multimodal And Multiobjective Optimization Using Real-coded Genetic Algorithms**" by *Amarendra Kumar* has been carried out under my supervision and it has not been submitted elsewhere for a degree.

Kalyanmoy Deb

Dr. Kalyanmoy Deb
Assistant Professor
Department of Mechanical Engineering
IIT Kanpur

February, 1996



ACKNOWLEDGEMENTS

I derive esteemed pleasure in expressing my sincere gratitude to Dr. Kalyanmoy Deb for his invaluable guidance and cooperation during the course of this investigation.

I would like to thank my senior colleagues Mr. N. Srinivas and Mr. Ram Bhushan Agrawal for giving fillip to my technical understanding of the subject. I am also grateful for the financial support provided by Department of Science and Technology, New Delhi under grant SR/SY/E-06/93. My special thanks goes to all my colleagues in CAD lab.

Amarendra Kumar

ABSTRACT

Real-coded genetic algorithms (GAs) do not use any coding of the problem variables, instead they work directly with the variables. The main difference in the implementation of real-coded GAs and binary-coded GAs is in their recombination operators. Although, a number of real-coded crossover implementations were suggested, most of them were developed with intution and without much analysis. Recently, a real-coded crossover operator has been developed based on the search characteristics of the single-point crossover used in binary-coded GAs. This simulated binary crossover (SBX) operator has been found to work well in many test problems having continuous search space compared to existing real-coded crossover implementations. The performance of real-coded GA with SBX has been investigated in solving multimodal and multiobjective problems. Sharing function approach and nondominated sorting implementations are included in the real-coded GA with SBX to solve multimodal and multiobjective problems, respectively. It is observed that the real-coded GAs perform equally well or better than binary-coded GAs in solving a number of test problems. One advantage of the SBX operator is that it can restrict children solutions to any arbitrary closeness to the parent solutions, thereby not requiring any separate mating restriction scheme for better on-line performance. Finally, real-coded GAs with SBX has been successfully used to find multiple Pareto-optimal solutions in solving an engineering welded beam design problem. These simulation results are encouraging and suggest the application of real-coded GAs with SBX operator to real-world optimization problems at large.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

Most real-world engineering design problems are associated with multiple solutions or multiple objectives. For a design engineer the knowledge of all alternate solutions is very important in decision making. A prototype may be fabricated with a number of solutions having the same objective value or it may have more than one objectives to be optimized simultaneously. Classical methods are not efficient in finding all these solutions in a single run.

Genetic Algorithms (GAs) has been found to work successfully in many science and engineering applications. In the simple genetic algorithms, all the points are converged to a single solution. In multimodal problems, the objective is to find multiple optimal solutions simultaneously. In order to coexist many optimal solutions in a population, Goldberg and Richardson (1987) developed *sharing functions.* In that study, the selection of individuals were done on the basis of an individual's shared fitness value. But due to unbiasedness in crossover operation, some lethal points are also created and solutions can be improved by applying a mating restriction scheme.

In classical methods, multiobjective problems are solved by combining all the objectives into a single objective. But the resulting solutions are dependent on weight factors or demand-level vectors. Moreover only one optimal solution can be achieved in single run. Schaffer (1984) tried to solve this by developing an algorithm VEGA. He used the basic recombination operators of GA in his algorithm, but could not get satisfactory results even after the addition of two heuristics. Motivated by Goldberg's concept, Srinivas and Deb (1994) devloped *Nondominated Sorting Genetic Algorithms*. NSGA differs with simple GA in selection of individuals and sharing between them.

The binary-coded genetic algorithms are successful in getting multimodal as well as multiobjective solutions in discrete search space. Recently a crossover and mutation operator have been developed by simulating the basic features of binary counterparts (Deb and Agrawal, 1995). The simulated binary crossover (SBX) operator in real-coded GA has a search power similar to the single-point binary-coded crossover operator.

This thesis examines the use of SBX in finding multimodal and multiobjective solutions. Chapter 2 describes the basic mechanics of simple binary-coded GA and related mathematical support. Chapter 3 presents the working procedure of SBX in continuous search space. Chapter 4 presents the comparision of binary-coded and real-coded GAs in finding multimodal solutions. To investigate further, this comparision is carried out on 50 randomly created bimodal functions. Further more, in order to reduce N^2 evaluations in sharing, the individuals of the test functions are shared with randomly selected individuals. Chapter 5 investigates the Pareto-optimal results achieved by both GAs in three test functions and a popular welded beam problem. Chapter 6 presents the further extensions. Chapter 7 presents the conclusion of the thesis.

Chapter 2

GENETIC ALGORITHMS : AN OVERVIEW

This chapter describes the basic mechanics and mathematical support of genetic algorithms. Genetic Algorithms (GAs) are artificial search techniques based on natural law of selection and genetics incorporating the Drawin's survival of the fittest principle.

2.1 Mechanics of Genetic Algorithms

In natural genetics, genes in the chromosomes act as a carrier for the qualities of individuals and their offsprings. In simple GA variables are coded in the form of strings containing *alleles* (either 1 or 0), similar to the *chromosomes* having genes in the biological system. In GA's terminology, strings along with their decoded values are known as *individuals*. A set of such individuals in the user-set range is called *population*. Like the natural genetics GA processes different population through *generation*. The three basic recombination operators of GAs are :

- Reproduction,
- Crossover, and
- Mutation.

The first operation employed in GAs is reproduction or selection. Based on alleles in the strings each individual has its decoded value or fitness value. In this selection process, individuals are given their representation or copies according to their fitness values in order to form a set of good individuals called *mating pool.* These individuals in the mating pool participate in the creation of children points.

One of the method of selection is *stochastic remainder proportionate selection.* In maxmization problem, using this method, probability of selection of an individual is $f_i / \sum f_i$, where f_i is the fitness value of the individual and $\sum f_i$ is the summation of the fitness of all the individuals present in the generation. Obviously an individual with a large f_i possesses a higher chance for its selection. This selection method is used for maximization problems. For minimization problem the function has to be suitably changed.

Tournament selection is another method of reproduction. This can be applied to both maxmization and minimization problems. In this method individuals are compred in tournaments. The number of individuals taken at a time for comparision is called the *tournament size.* In minimization problem with tournament size 2, two individuals are picked up at random and the individual with lower fitness value is given one copy. Generally the tournament size is kept 2.

The second operator in GA is crossover. Among the individuals in mating pool two are picked up at random and a random cross site is selected across the string length.

The alleles on one side of the site are swapped between the individuals to form two different strings called *children points*. For example, if s_1 and s_2 are two individuals and a random cross site is selected as shown:

$$\begin{array}{lcccc|ccc} s_1 & = & 1 & 0 & 1 & 0 & 0 & 1 \\ s_2 & = & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$$

After mating the two children points s_1' and s_2' will be

$$\begin{array}{lcccccc} s_1' & = & 1 & 0 & 1 & 1 & 0 & 1 \\ s_2' & = & 1 & 0 & 0 & 0 & 0 & 1 \end{array}$$

Crossover is the only operator which creates new individuals by mixing the genetic information of parent individuals. This operation is done with a prabability known as the crossover probability p_c.

Mutation, the third operator, plays a secondary role in GA. This is applied to get some features which are generally not achieved by crossover. Mutation is occasional alteration of an allele's value (changing 0 to 1 and vice versa) with some probablity called the mutation probability p_m. According to Goldberg (1989), mutation is an insurance policy against premature convergence. The action of the mutation operator is shown below :

$$1\ 0\ \boxed{1}\ 1\ 0\ 1 \xrightarrow{\text{Mutation}} 1\ 0\ \boxed{0}\ 1\ 0\ 1$$

A flow-chart of simple GA is shown in Figure 2.1. A mathematical treatment to the action of these operations is considered next.

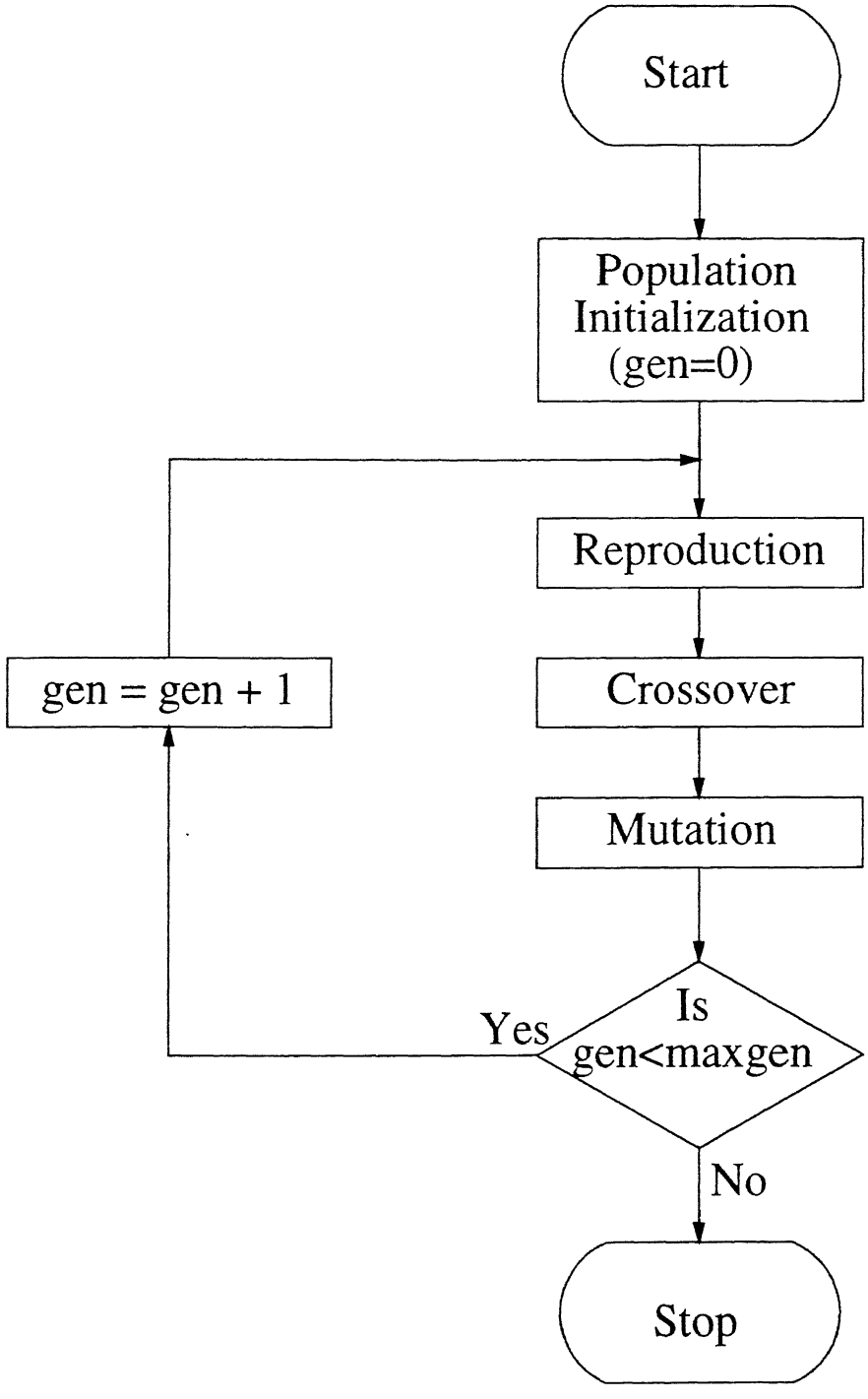


Figure 2.1: Flow-chart of a simple genetic algorithm (SGA)

2.2 Schema Theorem

The working principle of GA operators can be well understood by considering the processing of similiarities among the individuals. According to Holland (1975), a *schema (schemata,* plural) is a similiarity template describing a subset of strings at certain string positions. By introducing a "don't care" symbol, '$\star$', which can either take value 1 or 0, a schema can be expressed in terms of $\{0,1,\star\}$. Consider the following two strings.

$$\begin{array}{lcccccc} s_1 & = & 1 & 0 & 1 & 0 & 0 & 1 \\ s_2 & = & 0 & 1 & 0 & 0 & 1 & 0 \end{array}$$

Some of the schemata are shown below

$$\begin{array}{lcccccc} H_1 & = & \star & \star & \star & 0 & \star & \star \\ H_2 & = & \star & \star & 1 & 0 & \star & \star \\ H_3 & = & \star & 1 & 0 & \star & \star & \star \end{array}$$

s_1 is contained in H_1 and H_2 while s_2 is contined in H_1 and H_3. For a 6-bit string, there can be 3^6 (or 729) total schemata among 2^6 (or 64) unique strings.

There are two parameters which are used to describe a schema H more significantlly - *order* of a schema, $o(H)$, and its *defining length*, $\delta(H)$. The order of a schema is the number of fixed bits or positions (1's or 0's) present in the template while its defining length is the distance between the outermost defined positions in H. The order and defining length of the three schemata H_1, H_2, and H_3 are

$$\begin{array}{lcllcllcl} o(H_1) & = & 1 & o(H_2) & = & 2 & o(H_3) & = & 2 \\ \delta(H_1) & = & 0 & \delta(H_2) & = & 1 & \delta(H_3) & = & 1 \end{array}$$

According to Goldberg (1989), the expected number of a schema H in a population at generation t+1 may be computed by its known number $m(H,t+1)$ at generation t by the expression

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{f_{avg}}\left[1 - p_c\frac{\delta(H)}{l-1} - p_m o(H)\right],$$

where l is the string length, f_{avg} is the average fitness of the population. The fitness of the schema $f(H)$ is defined by

$$f(H) = \frac{\sum_{s_i \in H} f(s_i)}{m(H,t)}.$$

It is evident that a particular schema having low-order, short defining length and high fitness will receive increasing number of copies in forthcoming generations. The schema with these properties are called *building blocks*. According to Holland, GA with n population size processes n^3 schemata in parallel without accounting any extra memory.

The basic concept of GA search is the exponential allocation of increasing number of trials to many building blocks in parallel. The low-order building blocks form higher order schemata as the generation advances, thus leading to optimal or near optimal solutions. Due to these inherent features, GAs are robust and have been found to have an edge over the traditional optimization techniques.

2.3 Summary

The search procedures based on the mechanics of natural genetics and natural selection is the quintessence of genetic algorithms. The variables of the function are coded in

the form of strings and accordingly they get their fitness values from the objective function. GA has got three main operators namely : reproduction, crossover, and mutation. Reproduction is a selection process among individuals to form a mating pool of successful individuals. In crossover two individuals are chosen at random and a cross site is randomly chosen. The allele information on one side of the site is swapped between the mating individuals to create children points. Mutation tries to introduce some genetic information that is usually not possible by the crossover operator. The combinatorial performance of all the three operators has got a mathematical rapport as they exponentially allocate increasing number of trials to short, low-order schemata of good strings.

The next chapter deals with real-coded GAs with a newly developed crossover operator and a related mutation operator.

Chapter 3

REAL-CODED GENETIC ALGORITHMS

With the success of binary genetic algorithms in discrete search space and to eliminate the shortcomings, real-coded operators had been developed in continuous search space. In real-coded genetic algorithms variables are directly used, instead of coding them into binary strings. Davis (1991a) suggested to take the chlidren points as the average of parent points. Radcliffe's (1993) flat crossover randomly picks children points from the range set by parent points. Eshelman and Schaffer (1993) developed (BLX-α), but this was not developed from binary interval-schema processing point of view. None of the previous real-coded operators could match with that of binary GA . Recently Deb and Agrawal (1995) developed a real-coded crossover, *Simulated binary crossover*, (SBX) which has been found to perform equally well or even better than binary-coded GAs. The next sections cover the implementations of real-coded crossover and mutation.

3.1 Coding

In real-coded GA (RGA) using SBX, variables are directly initialized a real value by a random number u. For any function $f(x)$ to be optimized, the steps for the assignment of initial values to x are

- calculate a random number u,
- $x = (1-u) \star \text{low} + u \star \text{high}$.

Where low and high are the lower and upper limits of the variables, respectively. Thus after initialization we have directly the values of the variables to deal with. In binary GA, variables can acquire only the discrete values whereas RGA variables are free to accept any real value set by the users. Therefore RGA can search over the entire real space.

3.2 Simulated Binary Crossover (SBX)

In SBX, an effort has been made to simulate the salient features of a binary crossover operator. In binary crossover the children points are the mirror images of each other with respect to the mid-point of the parent individuals. Children points may be created inside the region of parent individuals (contracting crossover) or outside (expanding crossover) the region, without any biasedness in it. In fact, it all depends on the random cross site chosen across the string length of the mating individuals.

The SBX operator of real-coded GAs has been developed following the above mentioned properties. The children points are randomly created. If p_1 and p_2 are parent points and c_1 and c_2 are children points, a non-dimensionalized *spread factor* β has

been proposed as the ratio of the spread of created children points to that of parent points as follows:

$$\beta = \left| \frac{c_1 - c_2}{p_1 - p_2} \right|. \tag{3.1}$$

The parameter β can be calculated by the polynomial probability distribution as :

$$P(\beta) = \begin{cases} 0.5(n+1)\beta^n, & \text{if } \beta \leq 1 \\ 0.5(n+1)\frac{1}{\beta^{n+2}}, & \text{otherwise,} \end{cases} \tag{3.2}$$

where n is any nonnegative real number called the *distribution index*. A lower value of n gives a high probability of creating children points as distant points and vice versa. Thus, in RGA we have got a parameter n to handle the crossover for the creation of children points. Figure 3.1 shows the probability distribution as a function of β for different n values. Area of any curve for $\beta = 0$ to 1 is equal to 0.5 and from $\beta = 1$ to ∞ is equal to 0.5, so that the total area for $\beta = 0$ to ∞ is equal to 1 showing that the probabilty can have value between 0.0 and 1.0. The procedure to create children points from parent points using SBX is given as follows :

SBX Algorithm

- **Generate a random number u so that $0 \leq u \leq 1$**
- **Calculate the spread factor,**

$$\beta(u) = \begin{cases} (2u)^{\frac{1}{n+1}}, & \text{if u} \leq 0.5, \\ \frac{1}{2(u-1)}^{\frac{1}{n+1}}, & \text{otherwise.} \end{cases} \tag{3.3}$$

- **Calculate the children points.**

$$\begin{aligned} c_1 &= 0.5(p_1 + p_2) - 0.5\beta(u) \mid p_2 - p_1 \mid \\ c_2 &= 0.5(p_1 + p_2) + 0.5\beta(u) \mid p_2 - p_1 \mid \end{aligned} \tag{3.4}$$

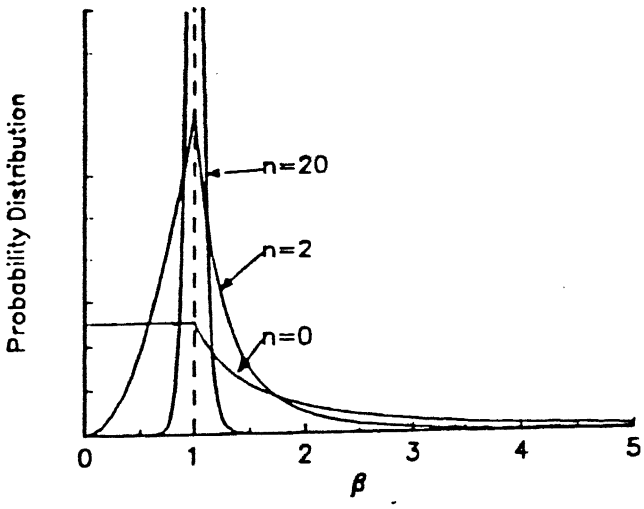


Figure 3.1: Probablity distribution of SBX

The random number generated, u, is actually the area of the curve between $\beta = 0$ to $\beta = \beta(u)$. For small range n=2 to 8, real-coded probablity distribution matches with SGA. In case of fixed variable bounds, the expanding probability distribution curve can be modified so that the probability of creating children points outside the limits is zero. The modified expanding probablity distribution is achieved by first calculating the cumulative probability

$$P' = \int_{\beta^{(L)}}^{\beta^{(U)}} P(\beta) d\beta, \tag{3.5}$$

and then using a modified probability $P(\beta)/P'$ instead of $P(\beta)$ to create children

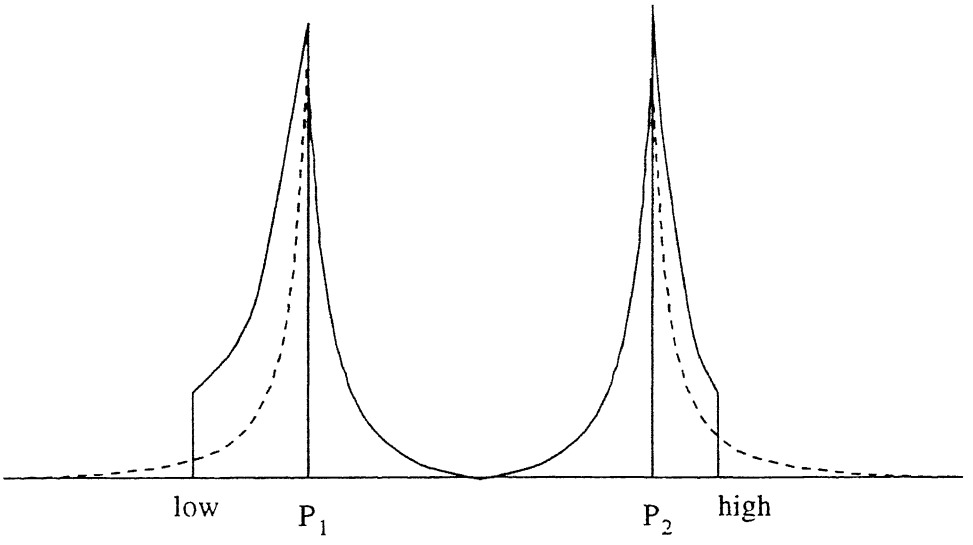


Figure 3.2: Probablity distribution for rigid bounds

points. In the above expression, the parameter $\beta^{(L)}$ and $\beta^{(U)}$ are the spread factor for the lower and upper bounds of the problem variable, respectively. Thus the modified probability distribution is helpful in an optimization problem which requires solutions to be found in the specified range only. A modified probablity distribution curve is shown in Figure 3.2. SBX can be separately applied to any number of variables .

3.3 Real-coded Mutation

In the binary mutation operation, we occasionally alter one of the alleles of a variable so that the decoded value of the string is perturbed. This can be achieved in RGA by taking a neighbouring point in the vicinity of current point. A *perturbance factor* δ can be defined as the ratio of actual perturbance Δ to maximum perturbance Δ_{max}

$$\delta = \frac{\Delta}{\Delta_{max}} = \frac{x_{new} - x_{old}}{\Delta_{max}}, \tag{3.6}$$

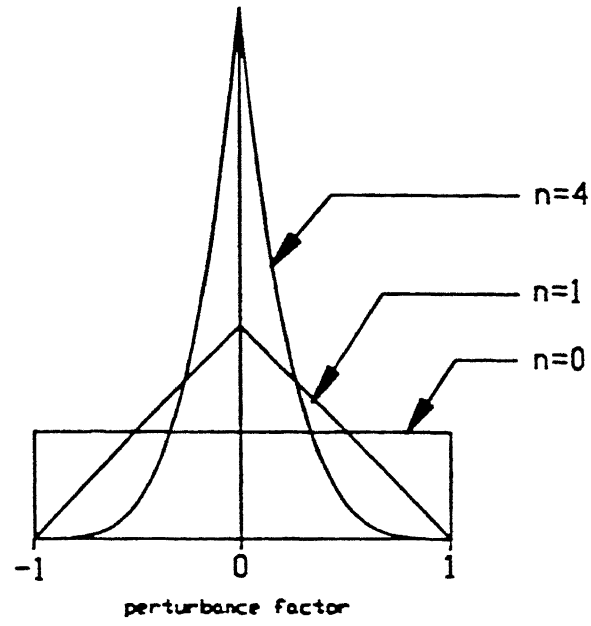


Figure 3.3: Mutation probablity distribution

where x_{old} and x_{new} are the values of the variable before and after mutation, respectively. This perturbance is applied using the following probability distribution :

$$P(\delta) = 0.5(n+1)(1-|\delta|)^n, \qquad -1 \leq \delta \leq 1, \tag{3.7}$$

where n is a non-negative number. The mutaion probability distribution has been shown in Figure 3.3. This distribution is similar to the contracting SBX except both parts of distribution are mirror images of each other. Equation 3.7 helps to find out the values of δ and the value of x_{new} is provided by the expression

$$x_{new} = x_{old} + \delta\Delta_{max}. \tag{3.8}$$

3.4 Summary

In real-coded opeartors, variables are directly given any real values between the lower and upper bounds of variables. Many real-coded crossovers exist but none of them could imbibe the inherent features of their binary counterparts. Deb and Agrawal (1995) developed real-coded Simulated Binary Crossover (SBX) which contains the fundamental features of a binary crossover. SBX works according to a probablity distribution curve and depending on the value of a random number generated it gives a value of spread factor. Relative posititons of children points with respect to parents depend on the value of spread factor. In case of rigid bounds, a modified probablity distribuiton curve is achieved by the division of the original probablity distribution and the cumulative probablity of finding the children inside the region. In real-coded mutation, the value of an individual is slightly altered by taking a point in the neighbourhood.

The next chapter deals with the successful implementation of RGA in multimodal function optimization.

Chapter 4

MULTIMODAL FUNCTION OPTIMIZATION

In multimodal function optimization problems, the objective is to find all the existing global or local optimal solutions at the same time. The information about multiple optimal solutions provides a better insight about the problem and is particularly useful to design engineers to choose an alternate solution, as and when required. The traditional optimization methods have to be applied a number of times to get different solutions because they are point-by-point search method. To achieve multiple optimal solutions, a particular method should start with a number of points and maintain it throughout the process. Theoretically this is possible in GA, as it works with a population. In the SGA, all the individuals are converged to a single peak. So there lies a need of allowing a number of schemata, representing different regions, to be processed and maintain simultaneously in subsequent generations. Goldberg and Richardson (1987) developed *sharing functions* and used it in the parallel investigation of many peaks.

4.1 Sharing Functions

Goldberg and Richardson (1987) tried to get multimodal solutions by 'gedanken' experiment on modified two-armed bandit problem. They observed that if the reproduction phase is carried out with a modified fitness obtained by degrading the original fitness value of a solution by a cumulative measure of the proximity between that solution and the rest of population, stable subpopulation can be maintained in the population. In natural terms it can be interpreted that an individual in a particular region has to share the resources around it in order to promote other individuals to move to unexplored regions or that with less number of individuals.

They defined a proximity measure (d_{ij}) between two individuals i and j either phenotypically (with the problem variables directly) or genotypically (with the corresponding strings). A sharing function $Sh(d_{ij})$ has been defined as follows:

$$Sh(d_{ij}) = \begin{cases} 1 - \frac{d_{ij}}{\sigma_{\text{share}}}, & \text{if } d_{ij} \leq \sigma_{\text{share}}; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

Where σ_{share} is the maximum distance upto which an individual has to share its objective function with the neighbours. The niche count m_i' for ith individual will be :

$$m_i' = \sum_{j=1}^{popsize} Sh(d_{ij}) = \sum_{j=1}^{popsize} Sh(d(x_i, x_j)), \tag{4.2}$$

where *popsize* is the population size. The shared fitness f_i' of an individual is defined as the ratio of potential fitness to the niche count m_i' :

$$f_i' = \frac{f_i'}{m_i'}. \tag{4.3}$$

If in a particular niche there is only one individual then $m_i' = 1$ and the individual will get its full objective value. This sharing concept is used only in reproduction, while

the other two operators remain unchanged. Here only the phenotypic sharing has been taken into consideration, as real-coded GAs uses the variables directly.

The calculations of the distance-metric and the sharing parameter have been described elsewhere (Deb 1989). For a single variable function having q peaks spread over the limits x_{min} to x_{max}, σ_{share} may be computed as

$$\sigma_{\text{share}} = \frac{X_{max} - X_{min}}{2q}, \tag{4.4}$$

where factor 2 takes care of sharing on either side of an individual. For a p dimensional space, the distance between two individuals is the Euclidian distance given by

$$d_{ij} = \sqrt{\sum_{k=1}^{p} (x_{k,i} - x_{k,j})^2}, \tag{4.5}$$

where $x_{k,i}$ and $x_{k,j}$ are the decoded values of i-th and j-th individuals, respectively. For a p-dimensional space problem, σ_{share} is calculated by assuming that each niche is enclosed in a p-dimensional hypersphere of radius σ_{share} so that each sphere encloses $\frac{1}{q}$ of the volume of whole space (Deb and Goldberg, 1989) :

$$\sigma_{\text{share}} = \frac{\sqrt{\sum_{k=1}^{p} (x_{k,max} - x_{k,min})^2}}{2\sqrt[p]{q}}. \tag{4.6}$$

For one variable prtoblem equation 4.6 simplifies to equation 4.4. If all the variables have unequal bounds, a normalized distance metric can be given by

$$d_{ij} = \sqrt{\sum_{k=1}^{p} ((x_{k,i} - x_{k,j})/(x_{k,max} - x_{k,min}))^2}, \tag{4.7}$$

so that the σ_{share} for the normalized distance metric can be estimated by

$$\sigma_{\text{share}} = \frac{1}{2\sqrt[p]{q}}. \tag{4.8}$$

4.2 Performance Measure

In order to characterize the distribution of population over the peaks, a chi-square-like criterion has been developed elsewhere (Deb, 1989), where the actual distribution is compared to the ideal distribution. For a q peak function, the expected number of individuals, μ_i, on i-th peak with fitness value f_i can be calculated as

$$\mu_i = \frac{f_i}{\sum f_i} \times N, \tag{4.9}$$

where N is the population size and $\sum_{i=1}^{N} \mu_i = N$. Based on the number of expected points on a peak and population size, the variance in the peak distribution will be

$$\sigma_i^2 = \mu_i \left(1 - \frac{\mu_i}{N}\right). \tag{4.10}$$

The expected number of points on non-peak region (μ_{q+1}) is zero and $\sum_{i=1}^{q+1} X_i = N$. It has been shown that

$$\sigma_{q+1}^2 = \sum_{i=1}^{q} \sigma_1^2. \tag{4.11}$$

Ideally there should not be any points on the non-peak region, but in practice some points are found on this peak also. A performance measure estimating the deviation of actual distribution of individuals (X_i) from the ideal distribution (μ_i) in all the (q+1) regions has been defined as

$$\text{performance, } \iota = \sqrt{\sum_{i=1}^{q+1} \left(\frac{X_i - \mu_i}{\sigma_i}\right)^2}. \tag{4.12}$$

The sharing scheme in a problem is considered to be working efficiently if the performance measure is near to zero (the ideal distribtion).

4.3 Sharing with Mating Restriction

The creation of children points depends on the two mating individuals. If they are from the same peaks, chances of creating children points outside the peak region are low and vice versa. Crossover operation picks two individuals which may be from the same peak or different peaks. In case of multimodal peaks, it is more likely to have mating individuals from different peaks.

The performance measure can be improved by reducing the generation of lethal individuals. A distance-metric similar to one used in the sharing function is used to identify the individuals from the same peak. A parameter σ_{mating} has been defined so that an individual can choose its mating partner around it up to the maximum σ_{mating} distance in the search space. In fixed population size there may be a case when an individual in unable to get a mating partner either because of all the individuals from the peak have already been selected or no one is present around it on the peak. In that case a random individual from the rest of the population is chosen as the mating partner. A flow-chart has been shown in Figure 4.1 to select two mating indivduals mate1 and mate2. Usually the mating distance is kept equal to σ_{share}.

4.4 Test Functions

In order to check the performance of GA with sharing in multimodal function optimization, different test functions as used in Deb (1989), are considered here. Four single-variable problems and one two-dimensional problem with four peaks have been tested. Fifty random functions with two equal peaks have also been created to confirm the sharing ability of binary as well as real-coded genetic algorithms.

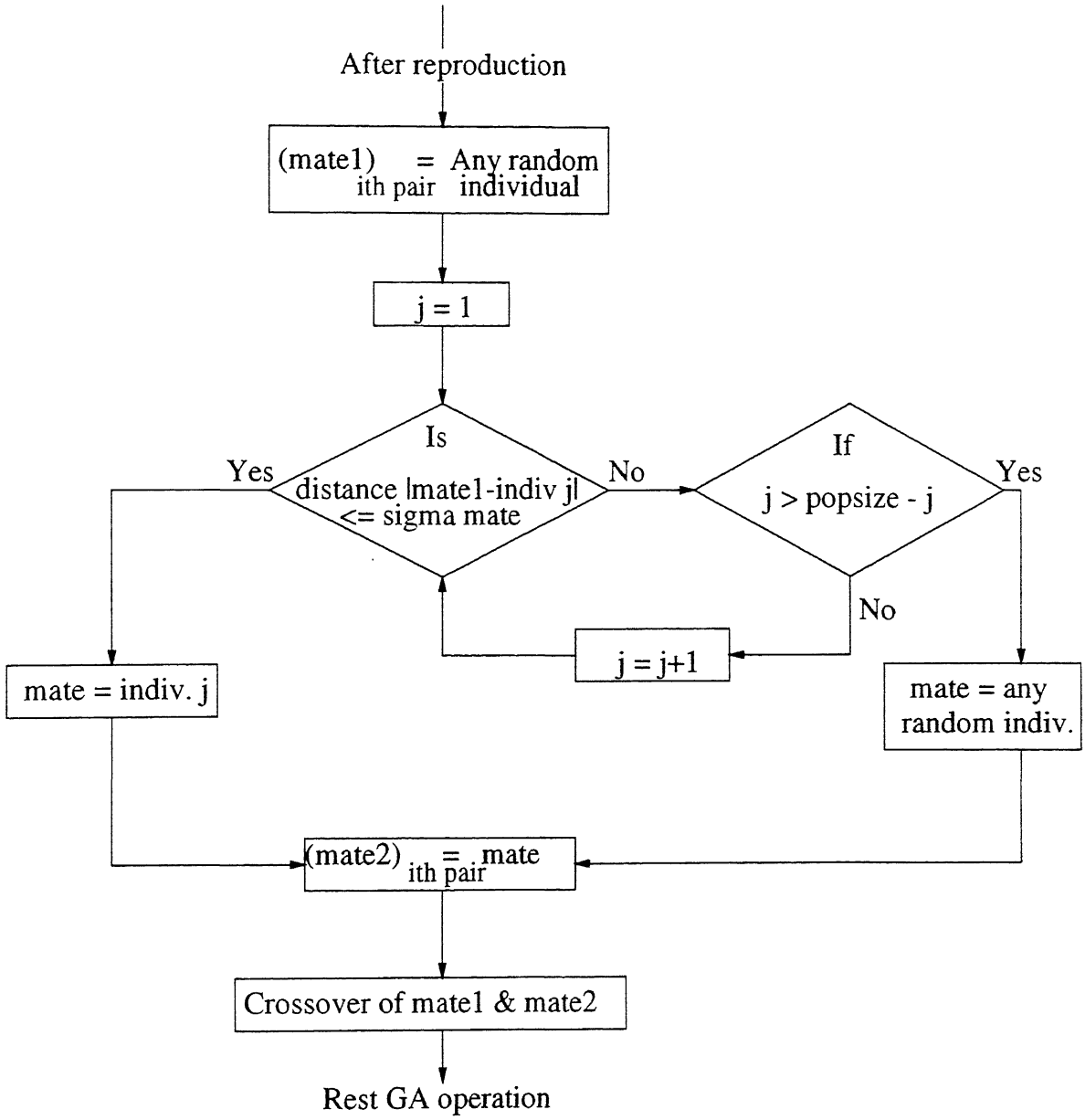


Figure 4.1: Flow-chart of sharing with mating restriction scheme

MM1 : A periodic function having peaks of equal size and interval

$$\text{Maximize} \quad f_{mm1}(x) = \sin^6(5\pi x).$$

This function has spread over $0 \leq x \leq 1$ with five equal peaks at equal intervals. The peaks are located at $x = 0.1$, 0.3, 0.5, 0.7, and 0.9. This is shown in Figure 4.2(a).

MM2 : A periodic function having peaks of unequal size and equal interval

$$\text{Maximize} \quad f_{mm2}(x) = e^{-2ln2\left(\frac{x-0.1}{0.8}\right)^2} \sin^6(5\pi x).$$

This function has also five peaks at equal intervals but the peaks are of unequal heights. The range of the function is $0 \leq x \leq 1$. The lengths of five peaks decrease exponentially and their heights are 1.0, 0.917, 0.707, 0.458, and 0.25, repectively. This has been shown in Figure 4.2(b).

MM3 : A periodic function having peaks of equal size and unequal interval

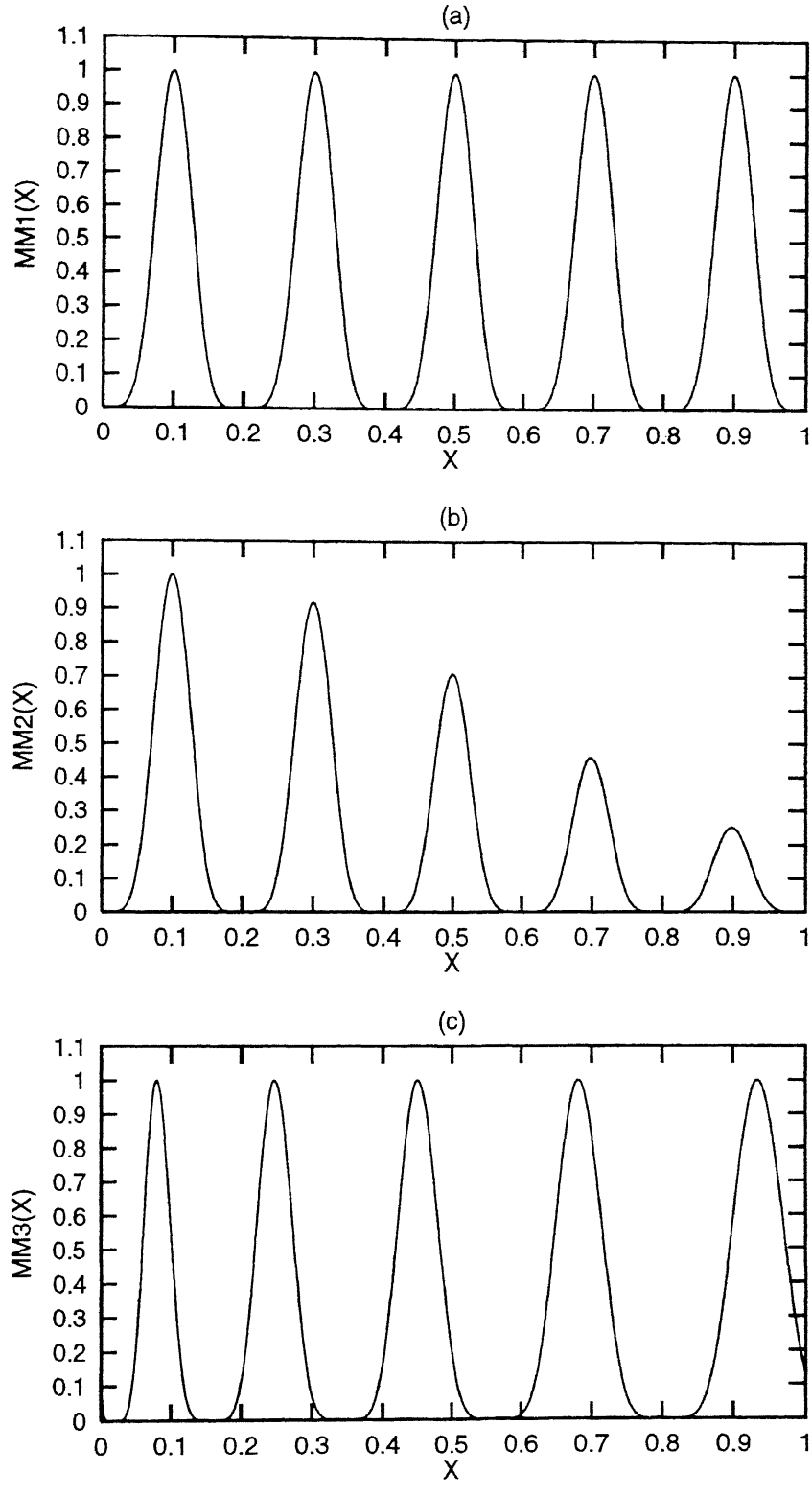
$$\text{Maximize} \quad f_{mm3}(x) = \sin^6(5\pi(x^{\frac{3}{4}} - 0.05)).$$

This function is having five equal peaks but located at unequal periods in the extreme limits $0 \leq x \leq 1$. The x values at different peaks are 0.08, 0.246, 0.45, 0.681, and 0.934 respectively. This is shown in Figure 4.2(c).

MM4 : A periodic function having peaks of unequal size and interval

$$\text{Maximize} \quad f_{mm4}(x) = e^{-2ln2\left(\frac{x-0.08}{0.854}\right)^2} \sin^6(5\pi(x^{\frac{3}{4}} - 0.05)).$$

This function is spreaded over the range $0 \leq x \leq 1$ with five peaks of unequal size located at unequal periods. The peaks fitness values decrease exponentially. This is shown in Figure 4.2(d).


(a)
MM1(X)
X
(b)
MM2(X)
X
(c)
MM3(X)
X

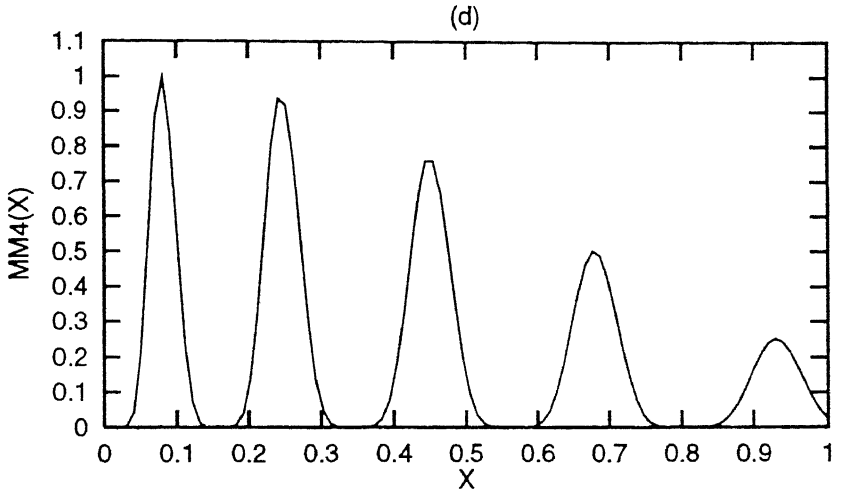


Figure 4.2: Functions MM1 - MM4

MM5 : Modified Himmelblau's function

$$\text{Maximize} \quad f_{mm5}(x) = \left[1 - \frac{(x_1^2+x_2-11)^2+(x_1+x_2^2-7)^2}{2186}\right]^{10}.$$

This function (Deb, 1989; Reklaites, Ravindran, and Ragsdell, 1983) has four peaks in the range $-6 \leq x_1, x_2 \leq 6$. This is shown in Figure 4.3.

MM6 : Fifty random functions with two equal peaks

$$\text{Maximize} \quad f_{mm6}(x) = \frac{\left[exp\left(-\frac{(x-x_1)^2}{2a^2}\right)+exp\left(-\frac{(x-x_2)^2}{2a^2}\right)\right]}{\left[1+exp\left(-\frac{(x_1-x_2)^2}{2a^2}\right)\right]}.$$

These functions have got two peaks at x_1 and x_2. The parameter a acts like the spread of the function. The above bimodal equations are obtained with 50 random combinations of $x_1, x_2 (\in (0,1))$ and $a \in (0, 0.05)$. One of the random function has been shown in the Figure 4.4.

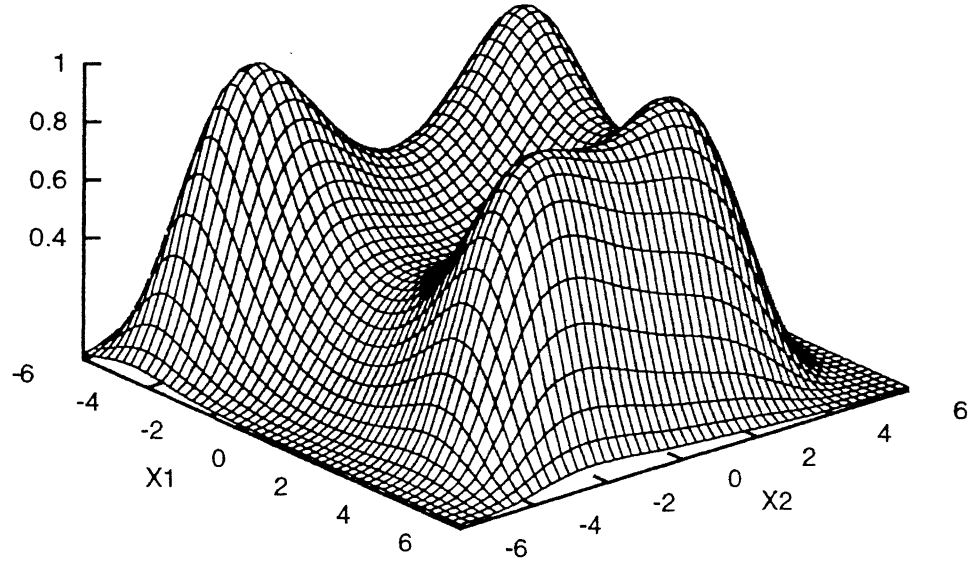


Figure 4.3: Himmelblau's function

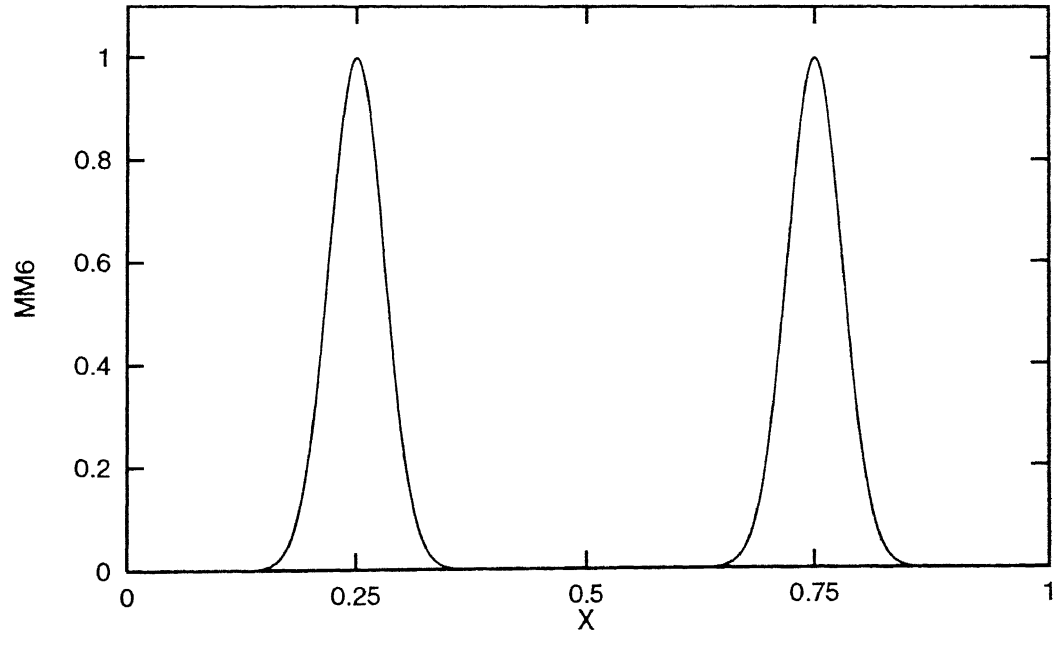


Figure 4.4: One of the random functions represented by MM6

4.5 Simulation Results

The simulation results of real-coded GAs are compared with binary coded GAs. The following GA parameters are used in all test functions MM1 - MM6 :

Maximum generation	200
Population size, N	100
String length, l	30
Crossover probability, p_c	0.9
Mutation probability, p_m	0.0
Distribution index, n	0 - 500

All the results have been taken in single run. Stochastic remainder selection is used here to minimize the stochastic error arising due to selection procedure.

4.5.1 Function MM1

Function MM1 has five uniform peaks at uniform intervals. All the results have been taken after 200 generatios, though the desired soultions are achieved much before. In simple binary-GAs without sharing, all the individuals are found to converge to one peak (Figure 4.5). Similar performance is shown by real-coded GAs for n=8 (Figure 4.6). In simple RGA a moderate value of n (2 to 8) provides good search power to the process. The value of sharing distance in this function is estimated by

$$\sigma_{\text{share}} = \frac{1.0-0.0}{2\times 5} = 0.1.$$

When sharing is applied, the individuals are found to accumulate on different peaks (Figures 4.7 and 4.8). In case of RGA a little higher value of n=35 is required to show similar binary sharing results. Higher value of n provides a steeper probability distribution curve, thereby allowing children points to be created near the parent points.

In sharing, some of the points are found on non-peak region also. This is due to the fact that crossover allows mating between any two random individuals. So there are chances that the children points may fall in non-peak region. Since solutions having a function values greater than 0.7 is considered to belong to the niche of an optimal solution, the difference between a child solution from the closest parent solution must be equal to 0.022. Thus, for contracting crossovers, the minimum spread factor must be equal to $\beta = 0.945$. If we want to succeed in 99% crossovers, the corresponding n can be approximately calculated by equating the cumulative probability of success to 0.99, as follows:

$$1 - \beta^{n+1} = 0.99.$$

For the two extreme optimal solutions, the above equation demands $n \approx 80$ and for two nearest optimal solutions ($x = 0.1$ and $x = 0.3$) the required distribution index is $n \approx 18$. The value of $n = 35$ compromise to the above two values of n.

Mating restriction scheme provides a safe selection of mating partner to reduce the chances of their offsprings to fall in non-peak region. The results with mating restriction scheme have been shown in Figures 4.9 and 4.10. In order to check the distribution ability of sharing scheme in binary as well as real GAs, performance measure in each generation has been calculated. For function MM1 the expected number of points in each peak is

$$\mu_i = \tfrac{1}{4} \times 100 = 25; \; \mu_6 = 0.$$

The variation of individual distribution in each peak is $\sigma_i^2 = 100 \times 0.2 \times 0.8 = 16$ so that $\sigma_6^2 = \sum_{i=1}^{5} = 5 \times 16 = 80$. The results of both GAs has been shown in Figure 4.11. RGA with n=35 shows similar results as that of binary-GA. The performance measure

of RGA with n=200 has also been shown in Figure 4.10. The average performance measures between generation 100 to 200 for different vaues of n (0 to 500) have been shown in Figure 4.12. This shows that for n=150 onwards, RGA results are better than binary GAs with mating restriction. Thus, in RGA, there is no need to use mating restriction scheme seperately. This can be achieved with higher values of n.

4.5.2 Function MM2

When simple binary-coded and real-coded GAs are applied to this function, all the individuals are converged to one peak (Figures 4.13 and 4.14). Since this function has five peaks in the same range as function MM1, the sharing distance is kept equal to 0.1. The expected number of individuals and the variance of the individuals on each peak is shown in Table 4.1.

Region	f_i	μ_i	σ_i^2
1st peak	1.000	30	21.00
2nd peak	0.917	27	19.95
3rd peak	0.707	21	16.72
4th peak	0.458	14	11.80
5th peak	0.250	8	6.94
non-peak		0	76.47

Table 4.1: Expected number and variance of different peak for MM2

In sharing both GAs are able to maintain subpopulations on different peaks (Figures 4.15 and 4.16) with some points on non-peak also. RGA requires n=35 to give the same distribution as that of binary-GAs. These results have been further improved by applying the mating restriction scheme (Figures 4.17 and 4.18). The performance measures of binary-GA and RGA for n=35 and n=200 have been shown in Figure 4.19.

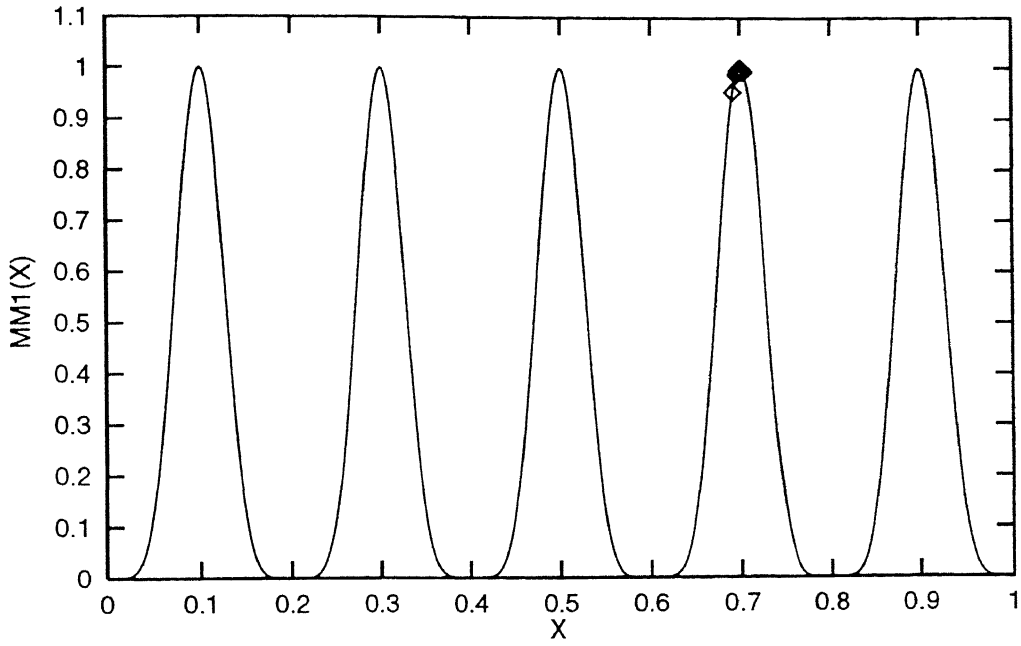


Figure 4.5: Individuals after 200 generations in binary-GA (no sharing) on MM1

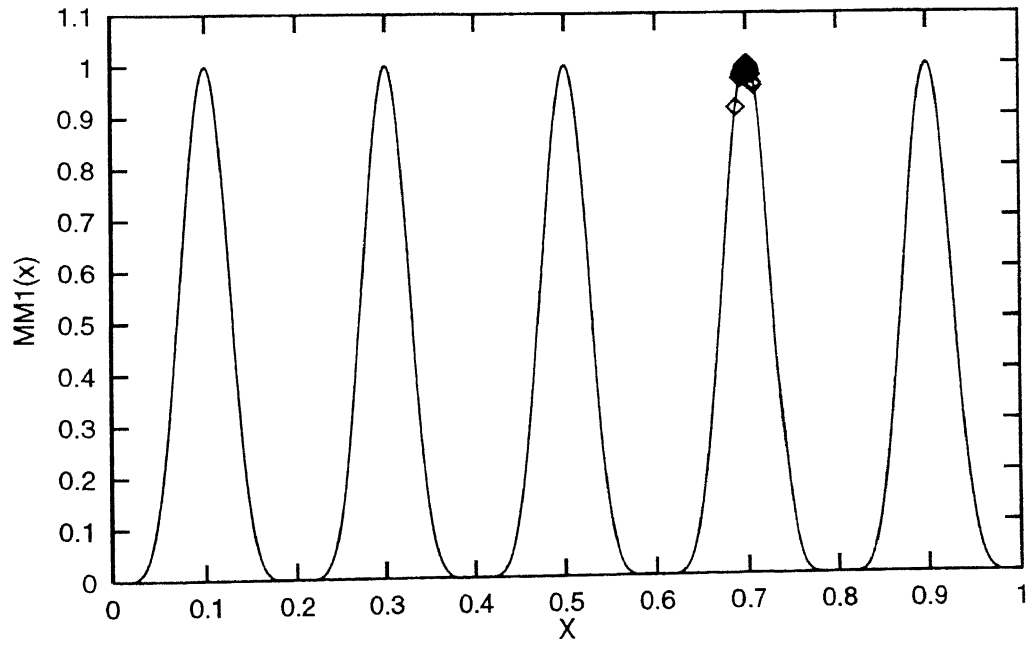


Figure 4.6: Points after 200 generations in real-GA (n=8, no sharing) on MM1

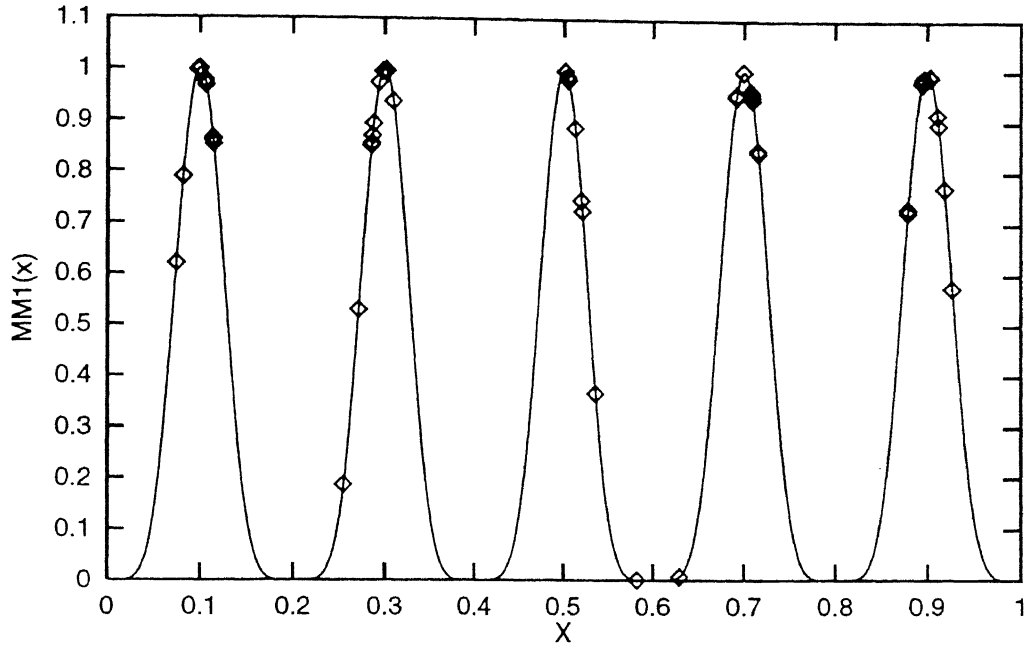


Figure 4.7: Individuals after 200 generations using sharing in binary-GA on MM1

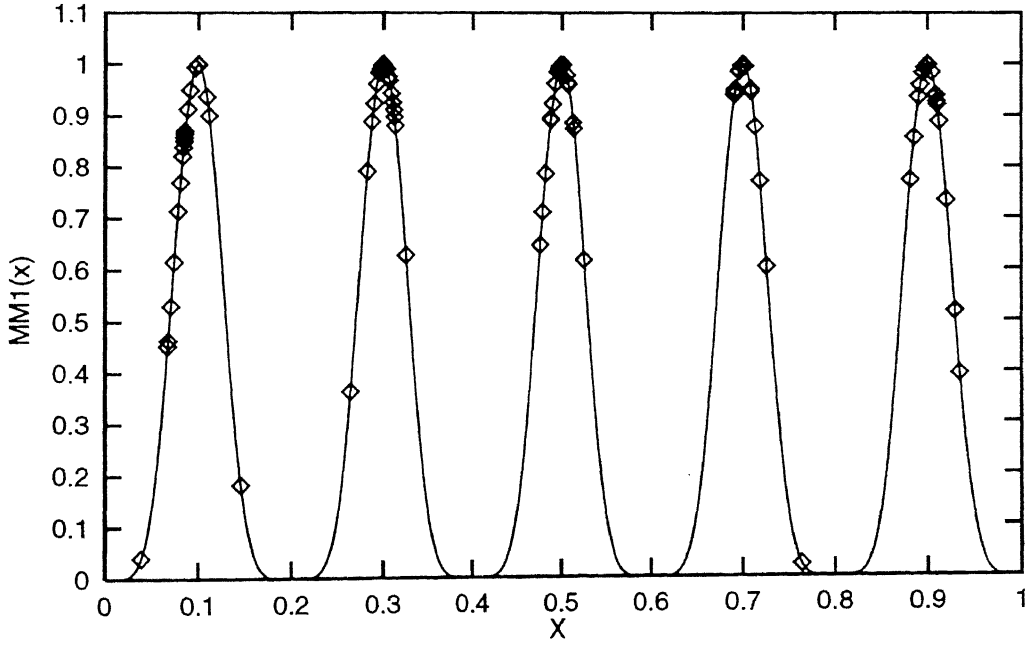


Figure 4.8: Individuals after 200 generations using sharing in RGA (n=35) on MM1

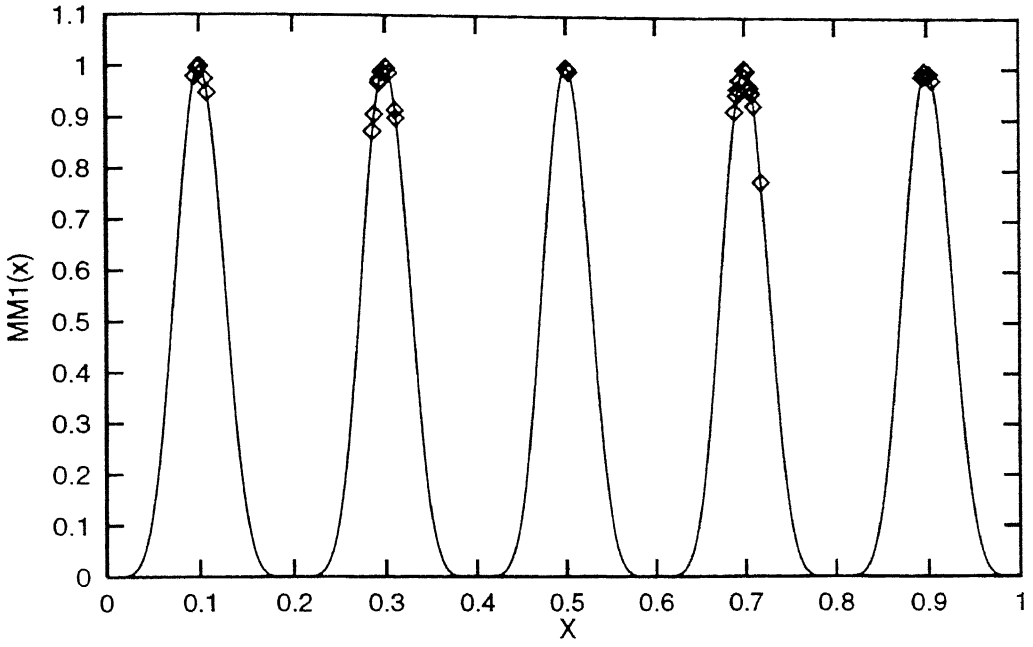


Figure 4.9: Points after 200 generations using mating restriction in binary-GA (MM1)

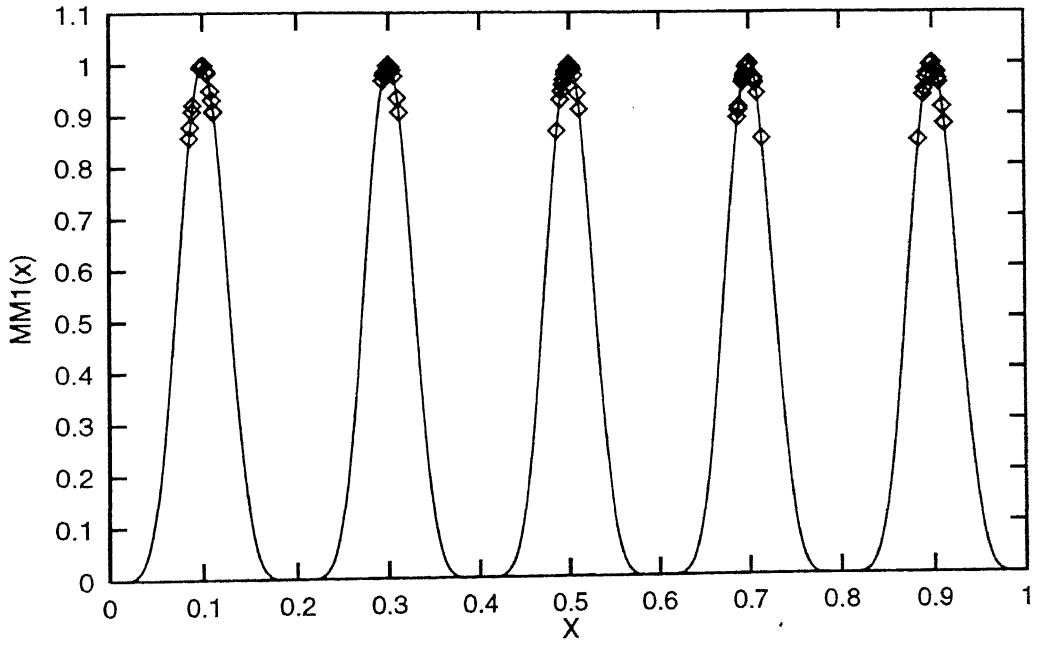


Figure 4.10: Points after 200 generations with mating restriction in RGA (n=35) on MM1

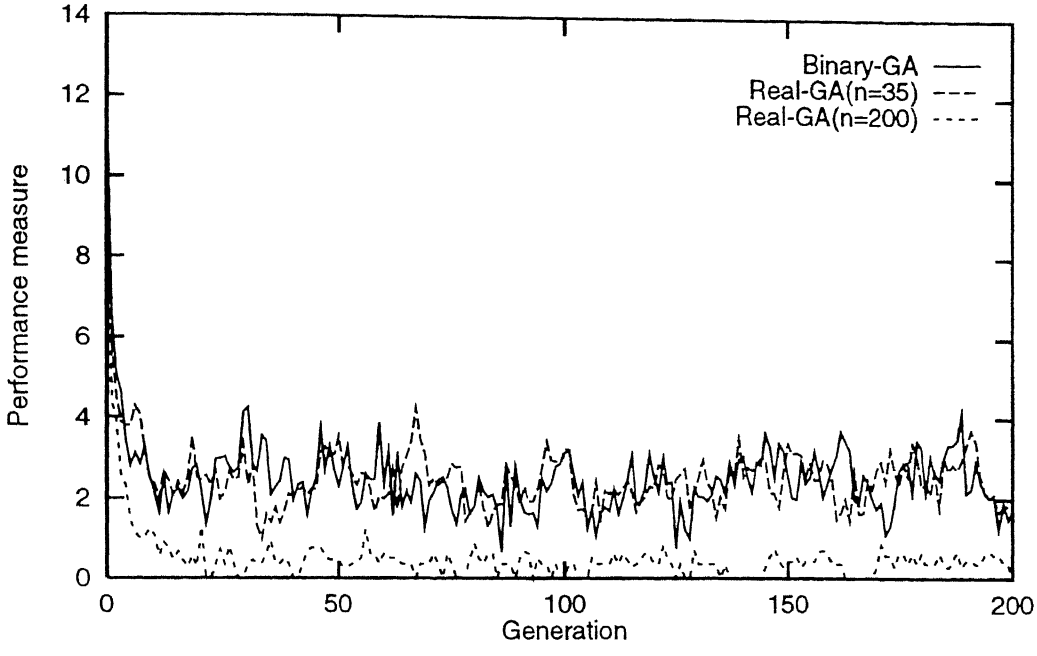


Figure 4.11: Performance measures of binary and real-coded GAs(Function MM1)

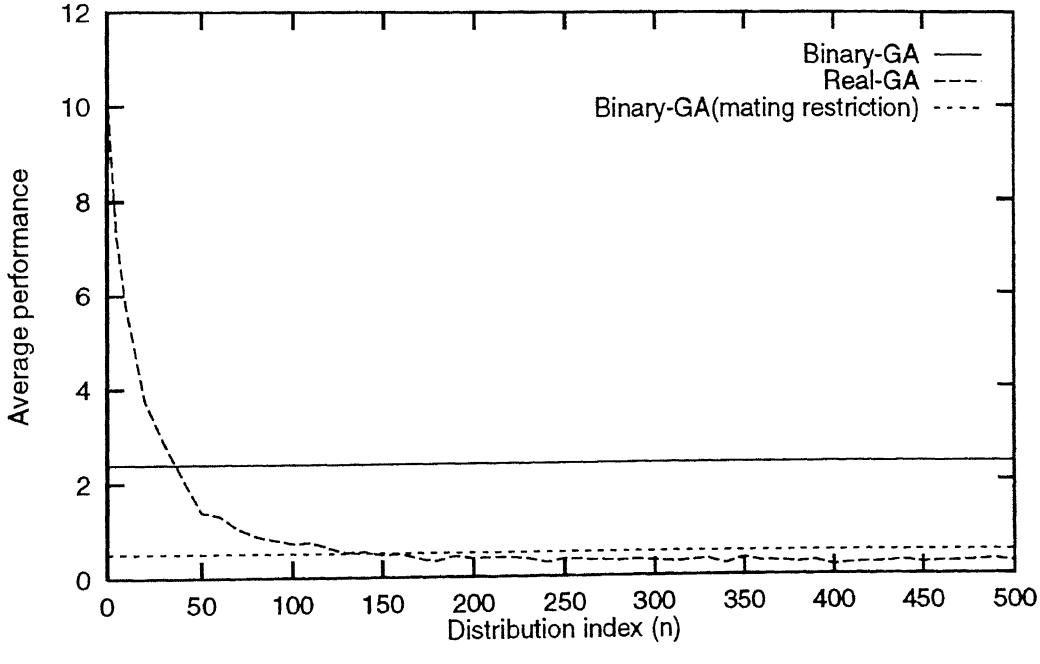


Figure 4.12: Average performance measures of binary and RGAs(Function MM1)

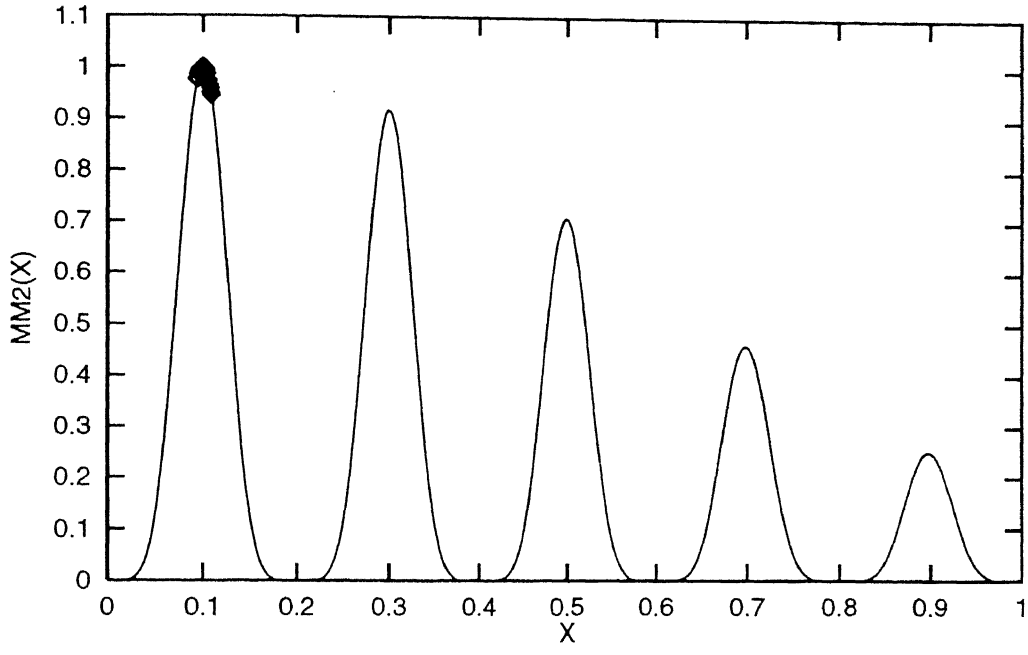


Figure 4.13: Individuals after 200 generations in binary-GA (no sharing) on MM2

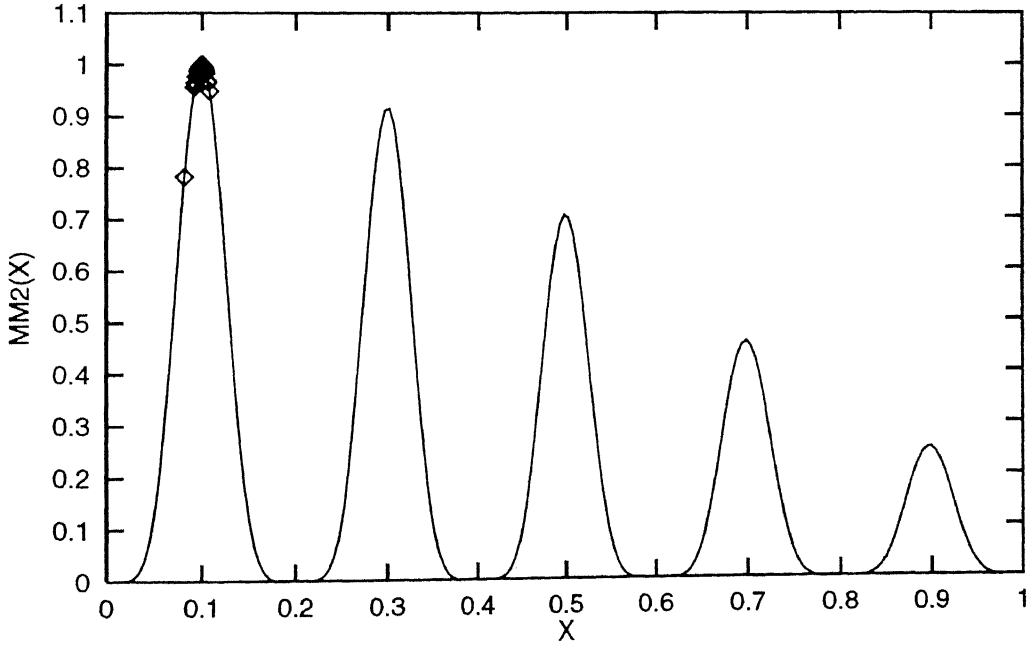


Figure 4.14: Individuals after 200 generations in RGA (n=8, no sharing) on MM2

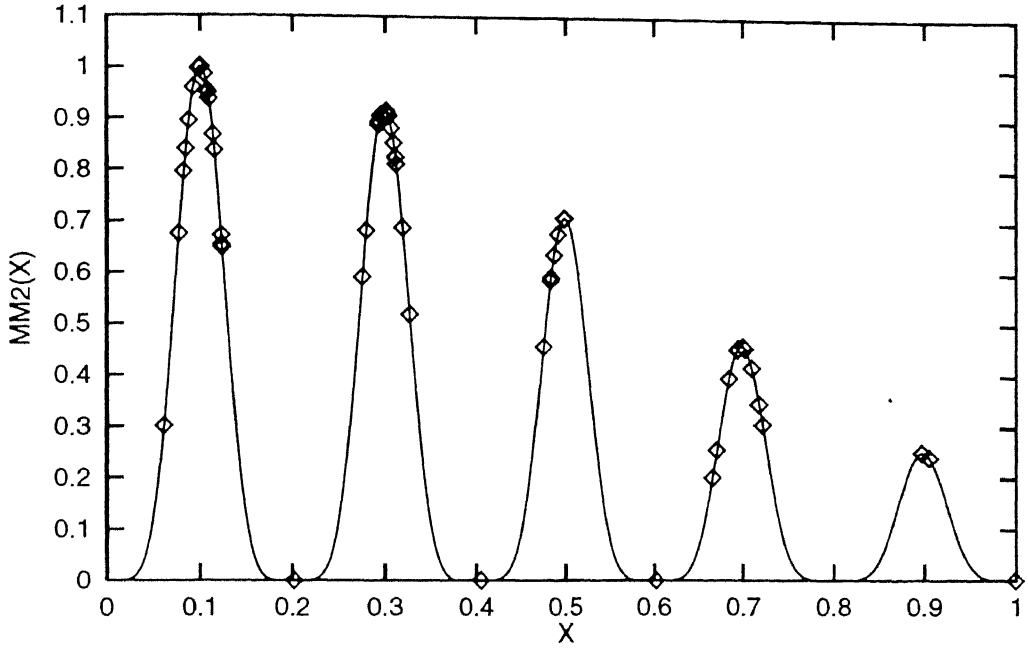


Figure 4.15: Individuals after 200 generations using sharing in binary-GA on MM2

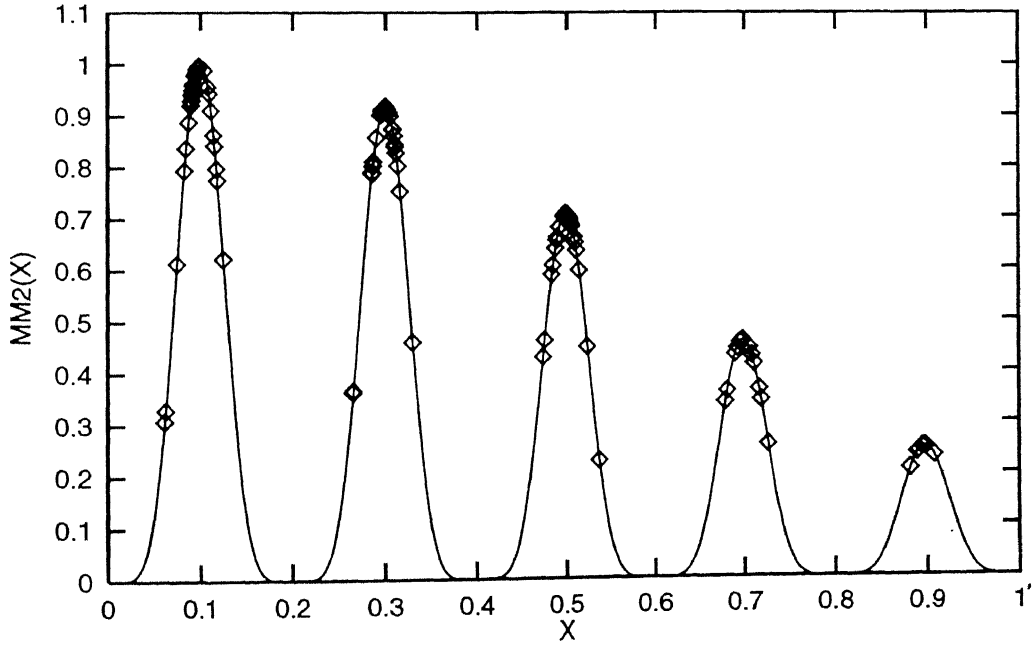


Figure 4.16: Individuals after 200 generations using sharing in RGA (n=35) on MM2

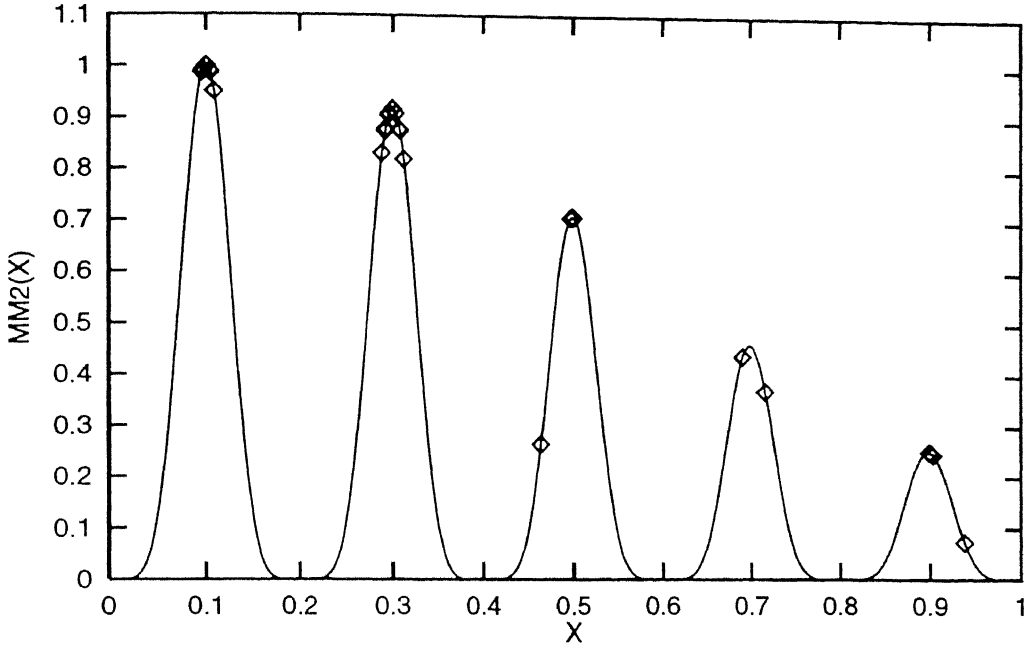


Figure 4.17: Points after 200 generations using mating restriction in binary-GA (MM2)

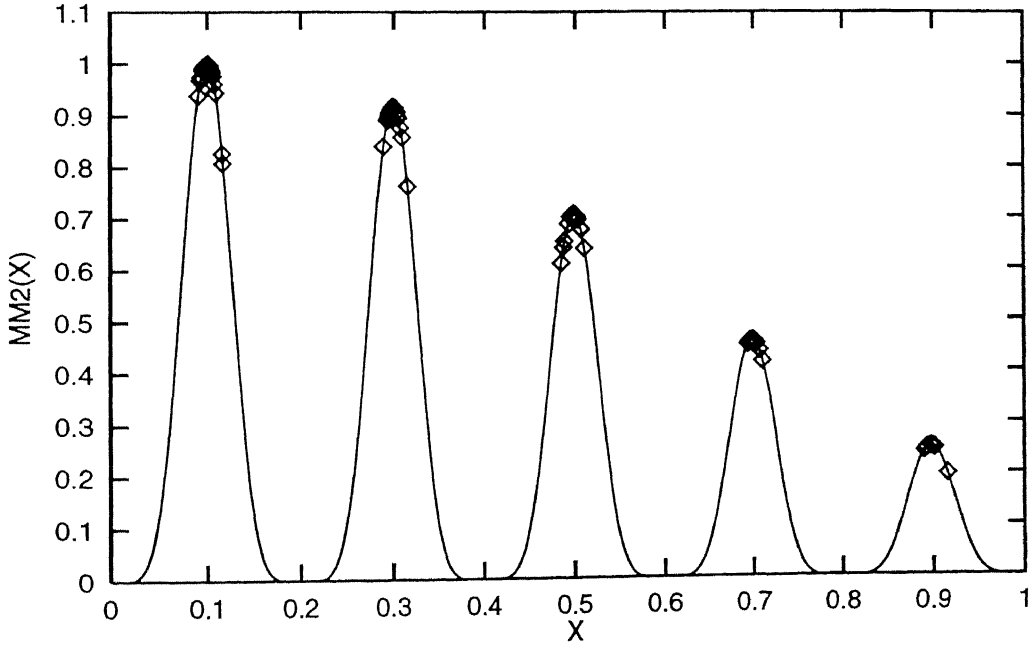


Figure 4.18: Points after 200 generations with mating restriction in RGA (n=35) on MM2

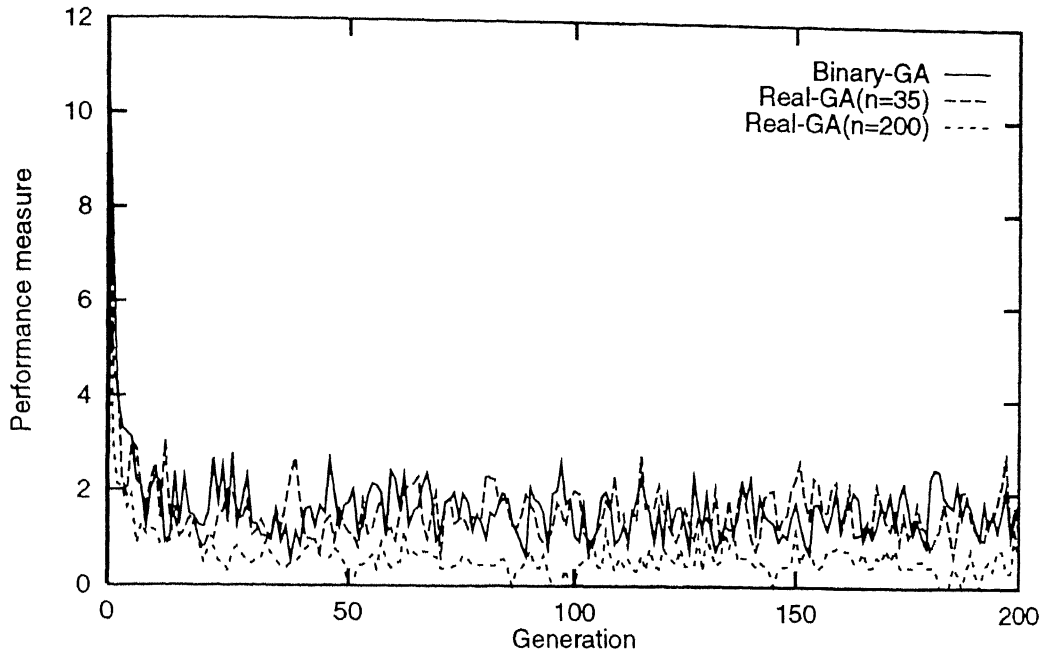


Figure 4.19: Performance measures of binary and real-coded GAs(Function MM2)

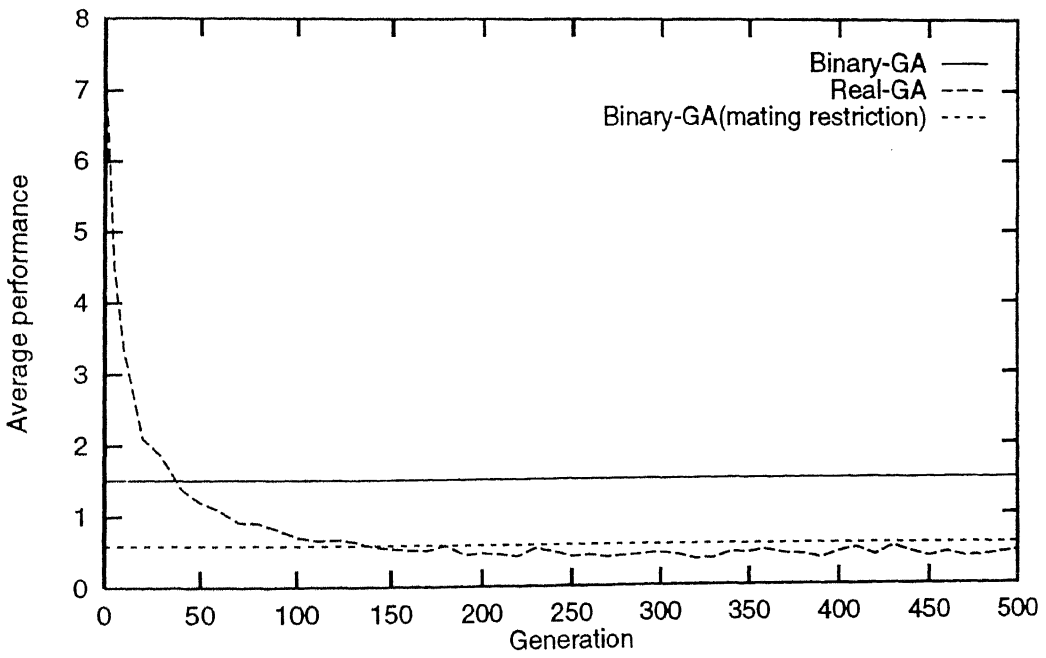


Figure 4.20: Average performance measures of binary and RGA (Function MM2)

The result of RGA with n =35 matches with that of binary-GA. When n=200 is used in RGA, the performance measures have been further improved. Figure 4.20 shows the comparision of average performance of RGA with that of binary-GA without and with mating restriction scheme. Average performance has been calculated from 100th to 200th generation. This result reveals that RGA with n=150 onwards is sufficient to produce better sharing results than that of binary-GA with mating restriction.

4.5.3 Function MM3

This function is having equal values of peaks located at unequal intervals. In simple binary as well as RGA, all the individuals converge on one peak (Figures 4.21 and 4.22). The value of n required in RGA is 8. Convergence of points on one peak depends on the chosen random seed number. In the sharing scheme, both GAs are able to maintain subpopulations on different peaks (Figures 4.23 and 4.24). Here the value of distribution index used in RGA is little higher (n=60) to get subpopulations on different peaks. Points on non-peak region are eliminated to a greater extent by using the mating restriction method (Figures 4.25 and 4.26). This function has equal number of expected points (μ_i=20) on each peak. The performance measures of binary and RGA (n=60) matches with each other, while that of RGA (n=200) is better than the later two results (Figure 4.27). Figure 4.28 shows that for n=160 and above the average performance (between $100th$ to $200th$ generation), RGA can substitute binary-GA with mating restriction. In this function also RGA with higher value of n behaves as if working with the mating restriction scheme.

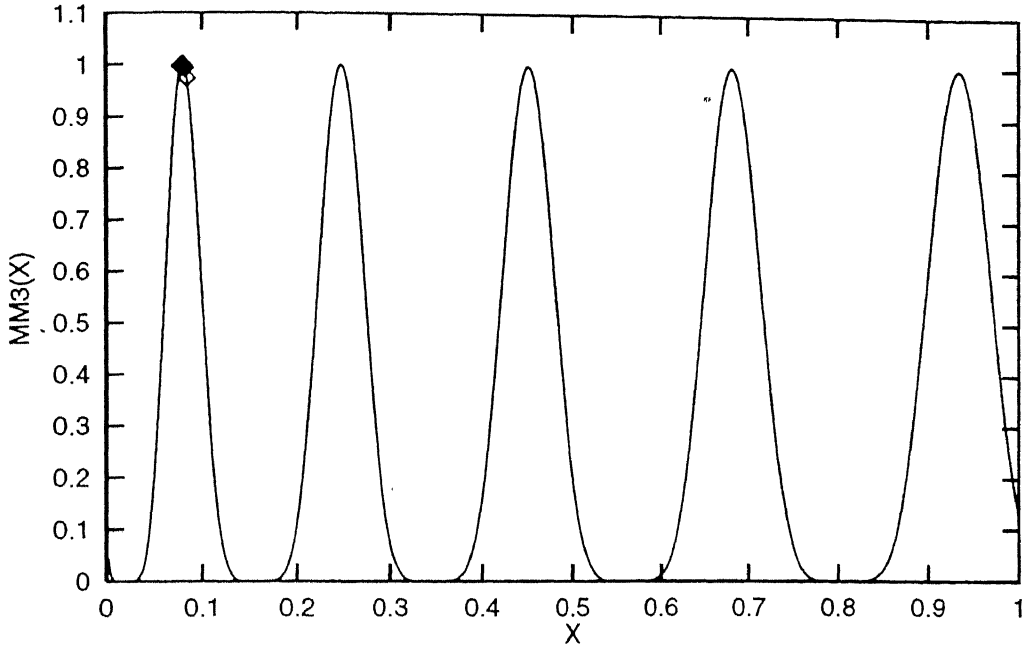


Figure 4.21: Individuals after 200 generations in binary-GA (no sharing) on MM3

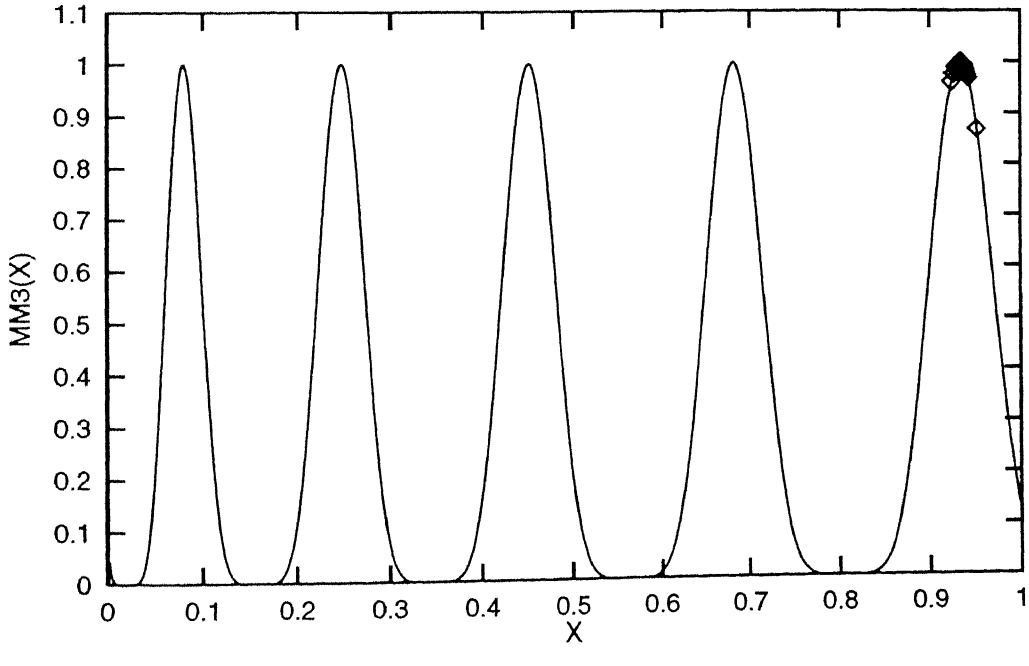


Figure 4.22: Individuals after 200 generations in real-GA (n=8, no sharing) on MM3

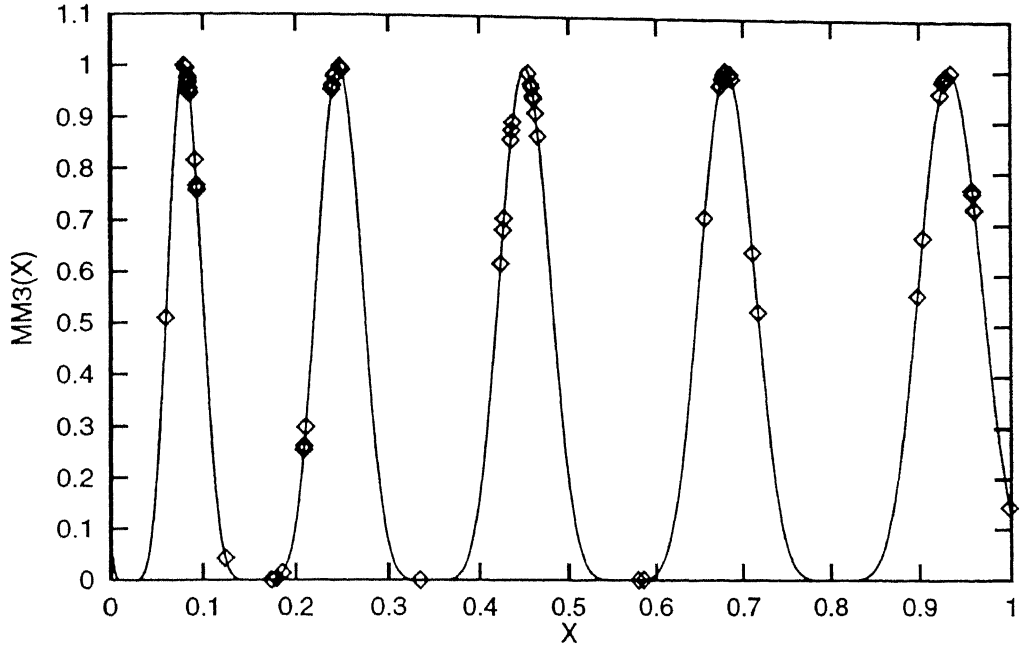


Figure 4.23: Individuals after 200 generations using sharing in binary-GA on MM3

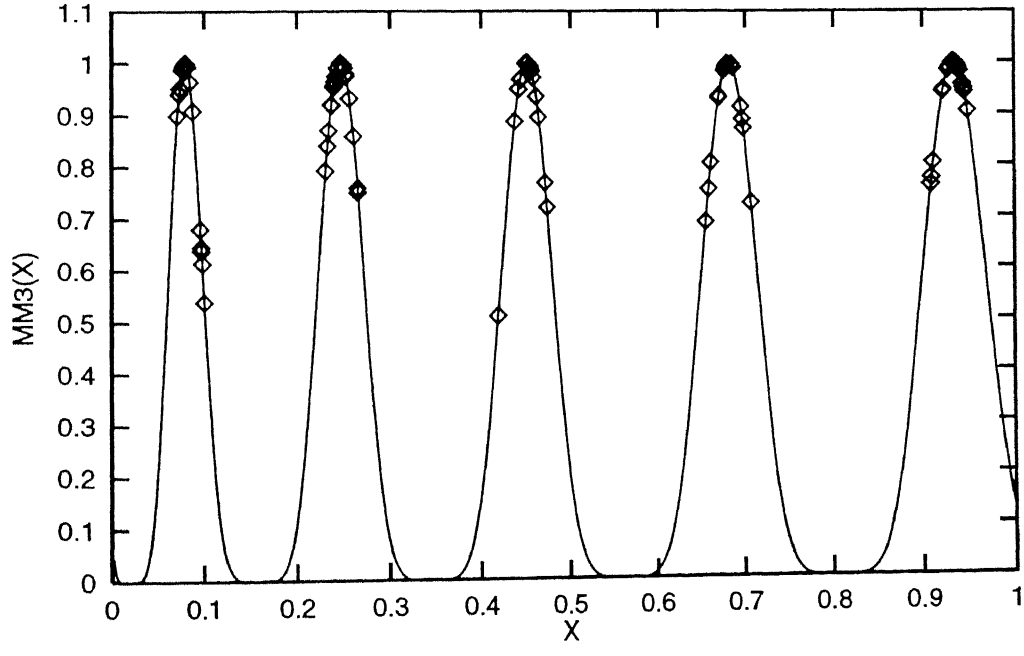


Figure 4.24: Individuals after 200 generations using sharing in RGA ($n = 60$) on MM3

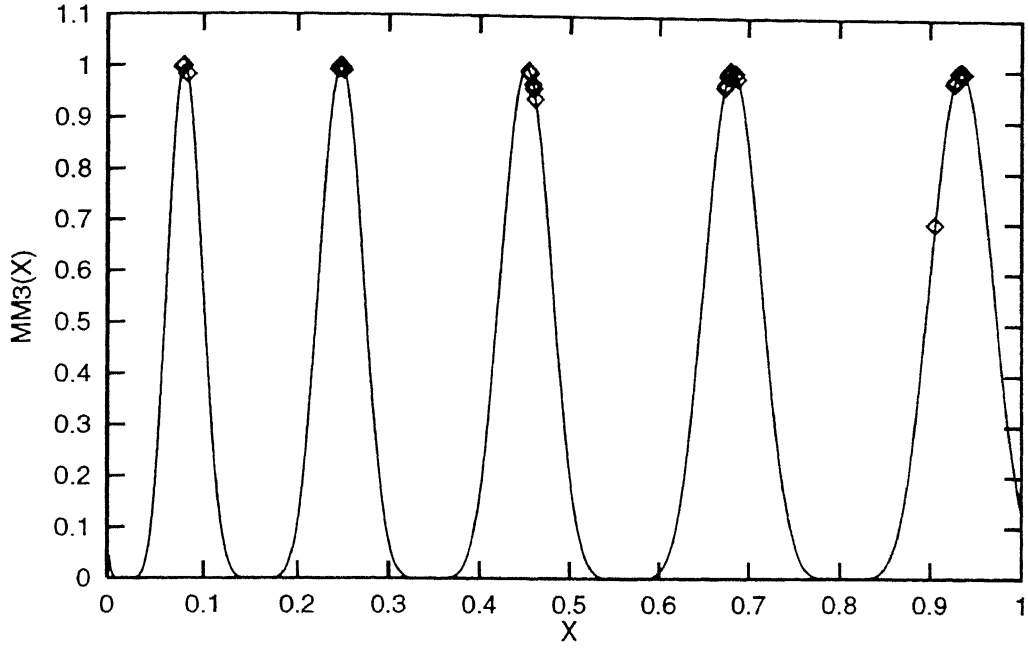


Figure 4.25: Points after 200 generations using mating restriction in binary-GA (MM3)

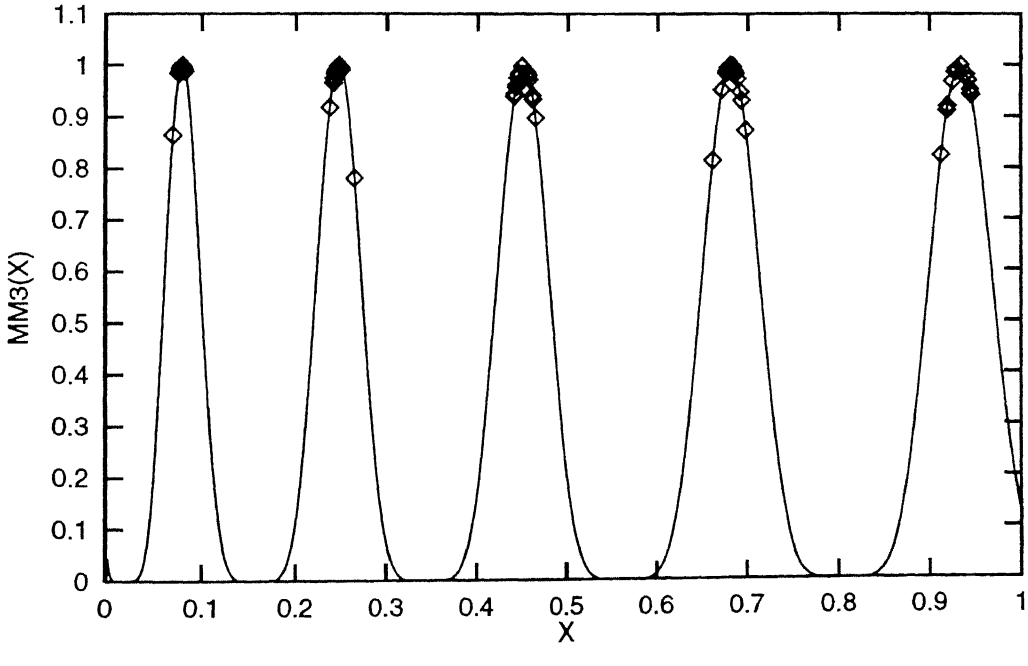


Figure 4.26: Points after 200 generations with mating restriction in RGA (n=60) on MM3

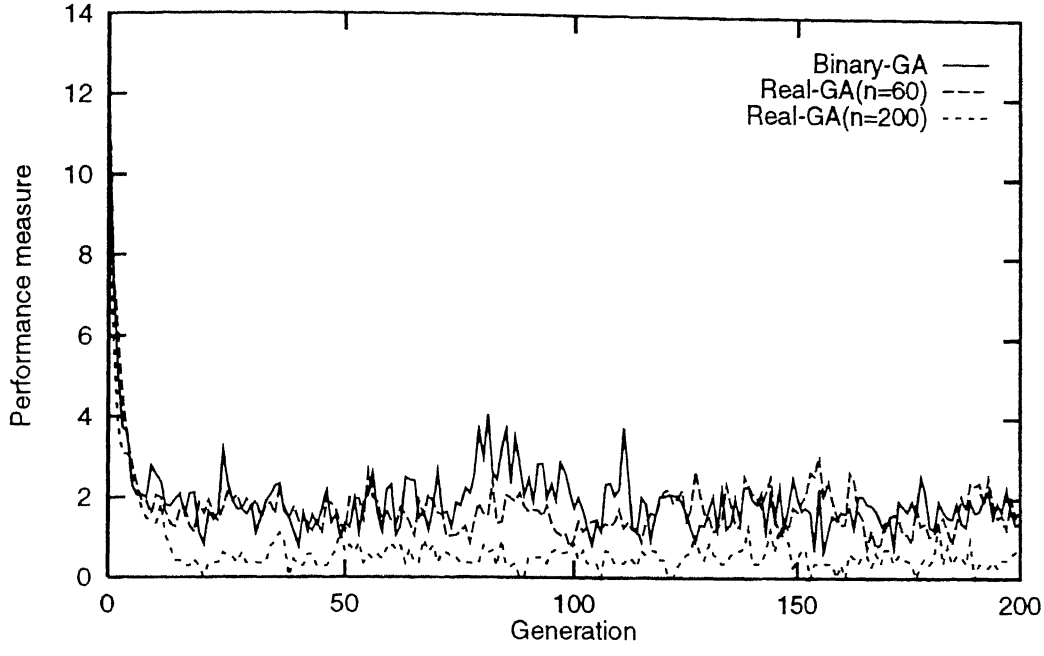


Figure 4.27: Performance measures of binary and real-coded GAs(Function MM3)

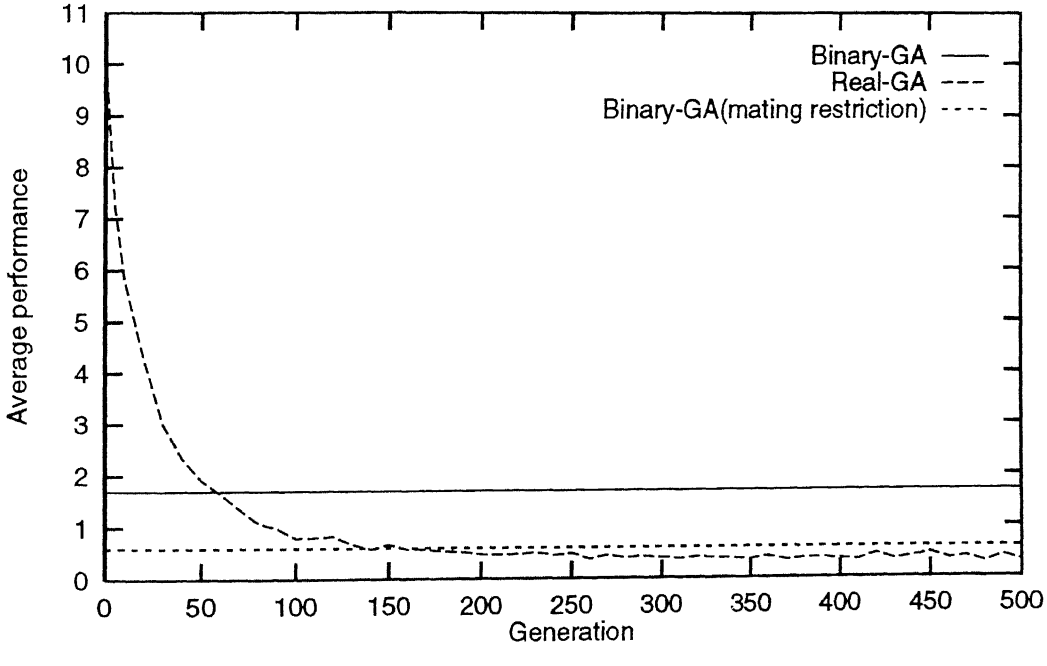


Figure 4.28: Average performance measures of binary and RGA (Function MM3)

4.5.4 Function MM4

Like binary-GA, RGA (n=8) is able to converge all individuals on one peak (Figures 4.29 and 4.30). When the sharing scheme is incorporated in both the GAs, they are able to diversify the population on different peaks (Figures 4.31 and 4.32). But the GAs are not able to restrict the individuals on non-peak region. This problem has been eliminated to much extent by using sharing with mating restriction scheme (Figures 4.33 and 4.34). Table 4.2 shows the number of expected points (μ_i) and variance (σ^2) of individuals on each peak. Figure 4.35 shows that the performance measures of both binary as well as RGA (n=35) are similar whereas RGA (n=200) shows comparatively good results. RGA with n=150 and above provides better results than binary-GA with mating restriction scheme (Figure 4.36).

Region	f_i	μ_i	σ_i^2
1st peak	1.000	29	20.52
2nd peak	0.950	27	19.88
3rd peak	0.770	22	17.27
4th peak	0.500	15	12.33
5th peak	0.250	7	6.68
non-peak		0	76.68

Table 4.2: Expected number and variance of each peak for MM4

4.5.5 Function MM5

This is a two parameter function with four equal-height peaks. Figures 4.37 and 4.38 illustrate the convergence of all the points on a single peak. The σ_{share} will be

$$\sigma_{\text{share}} = \frac{\sqrt{2 \times (6-(-6))^2}}{2 \times 4^{1/2}} = 4.24.$$

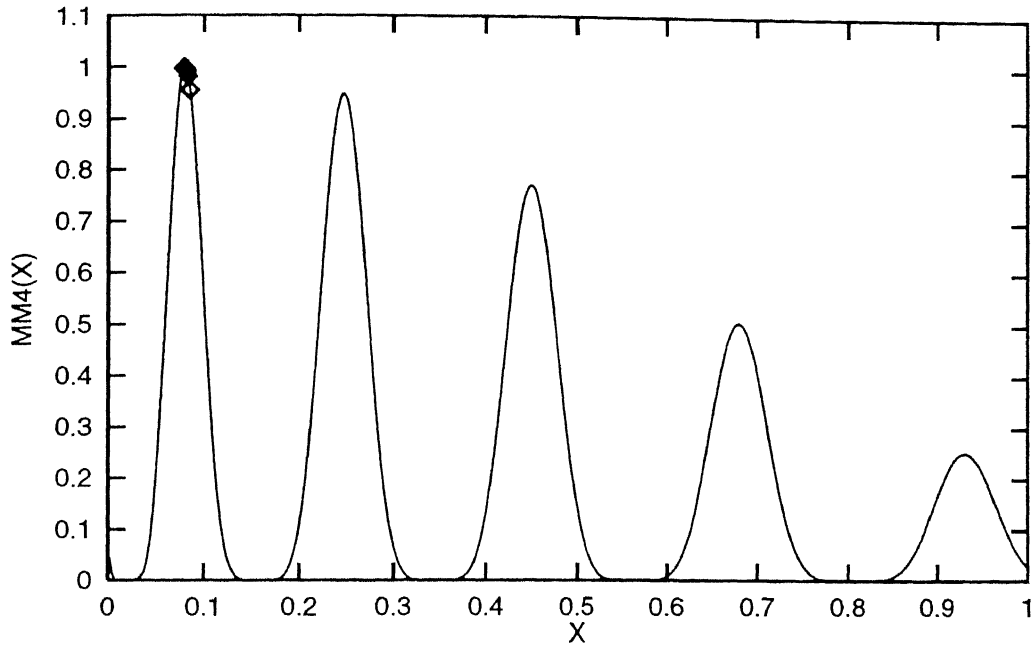


Figure 4.29: Individuals after 200 generations in binary-GA (no sharing) on MM4

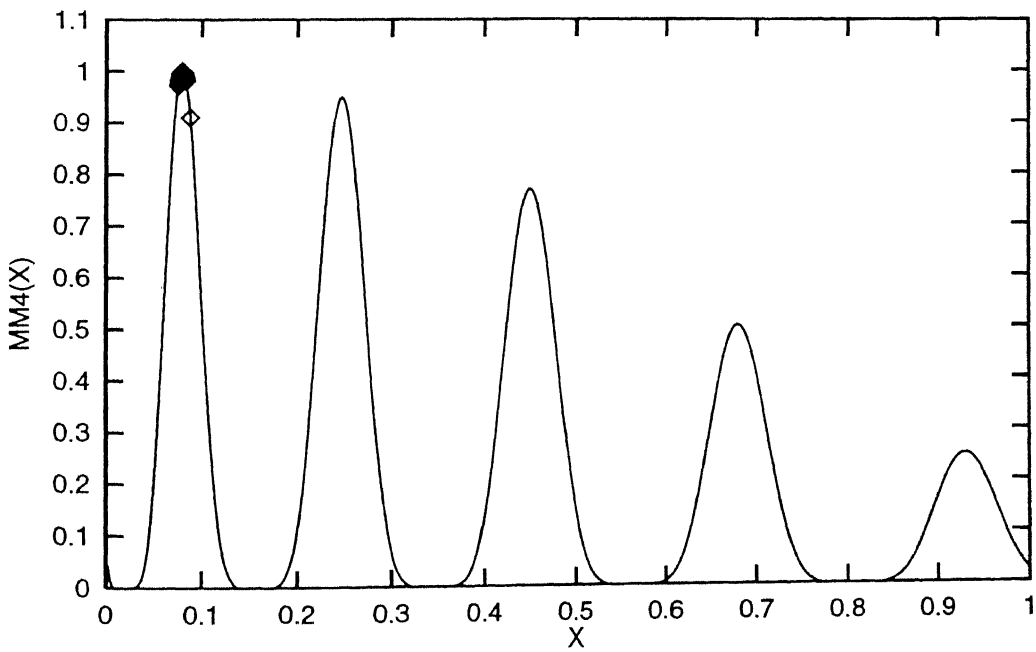


Figure 4.30: Individuals after 200 generations in RGA (n=8, no sharing) on MM4

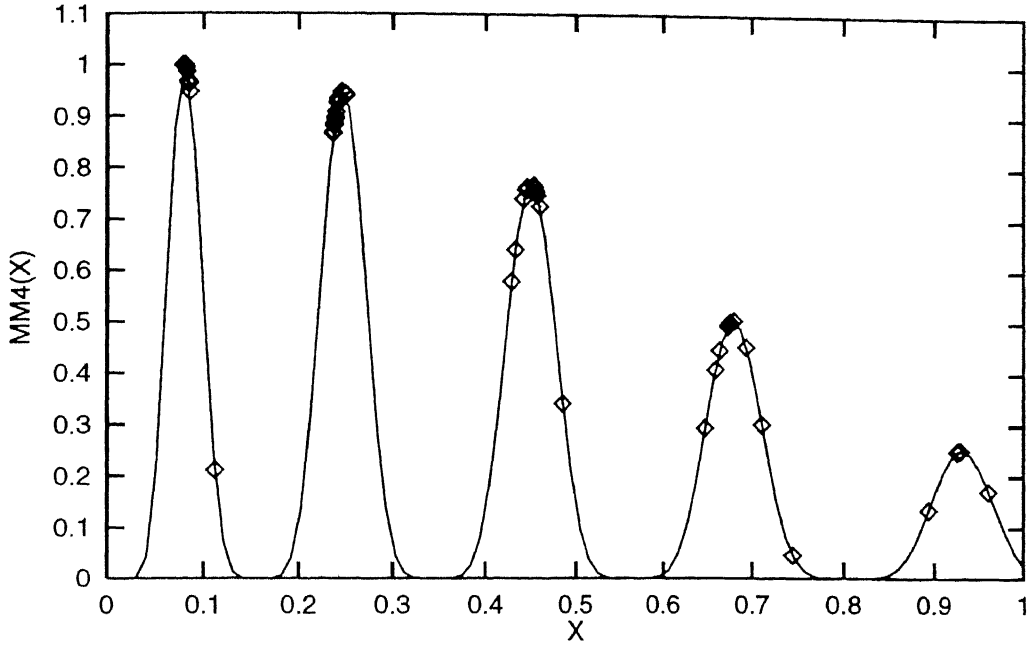


Figure 4.31: Individuals after 200 generations using sharing in binary-GA on MM4

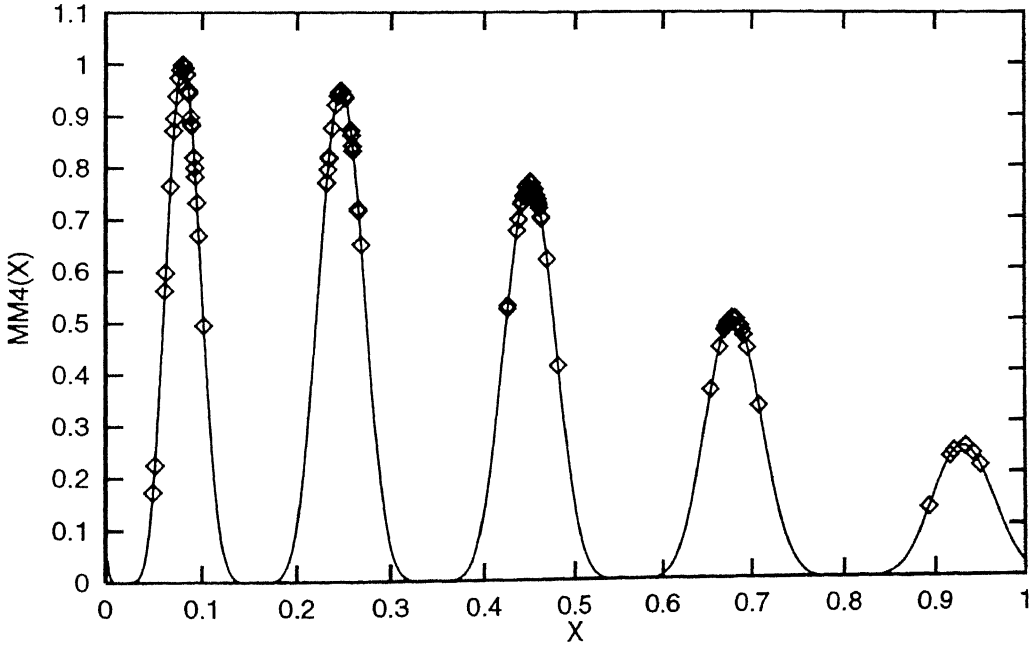


Figure 4.32: Individuals after 200 generations using sharing in RGA (n=35) on MM4

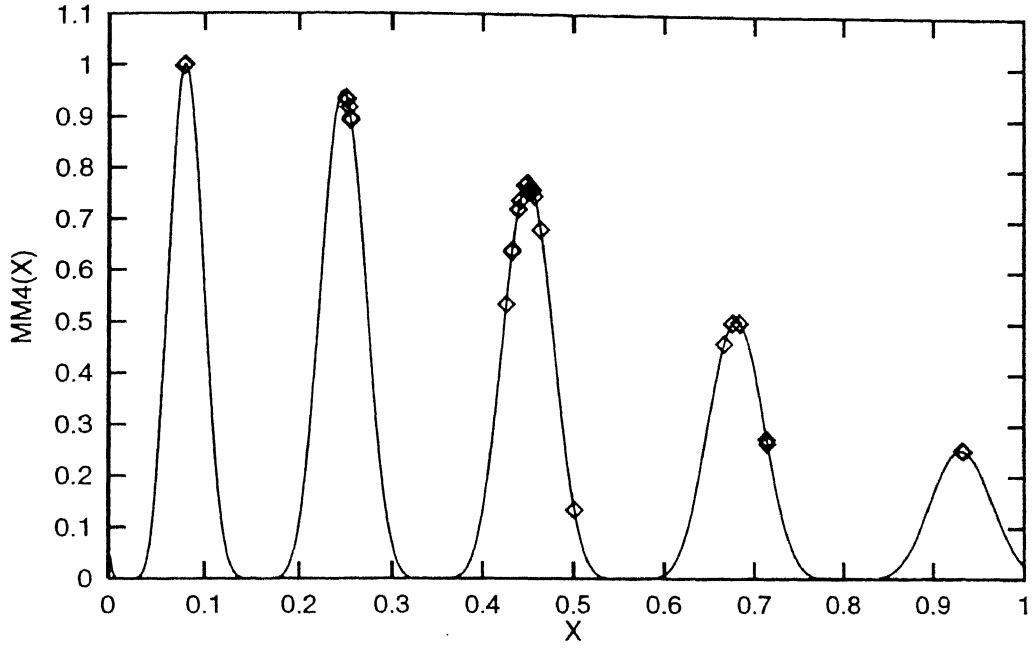


Figure 4.33: Points after 200 generations using mating restriction in binary-GA (MM4)

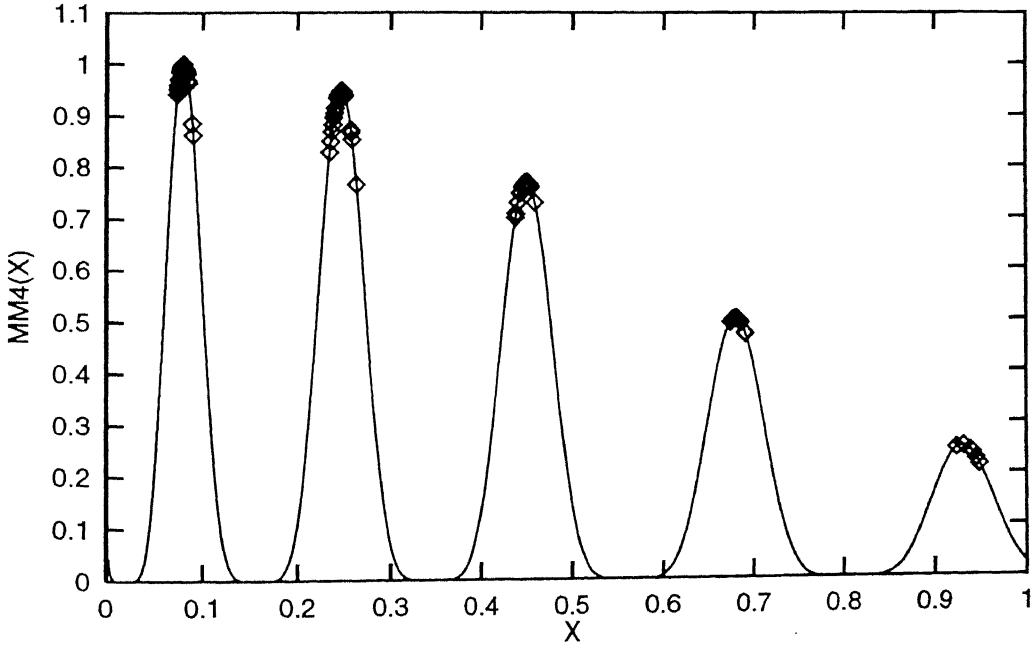


Figure 4.34: Points after 200 generations using mating restriction in RGA (n=35) on MM4

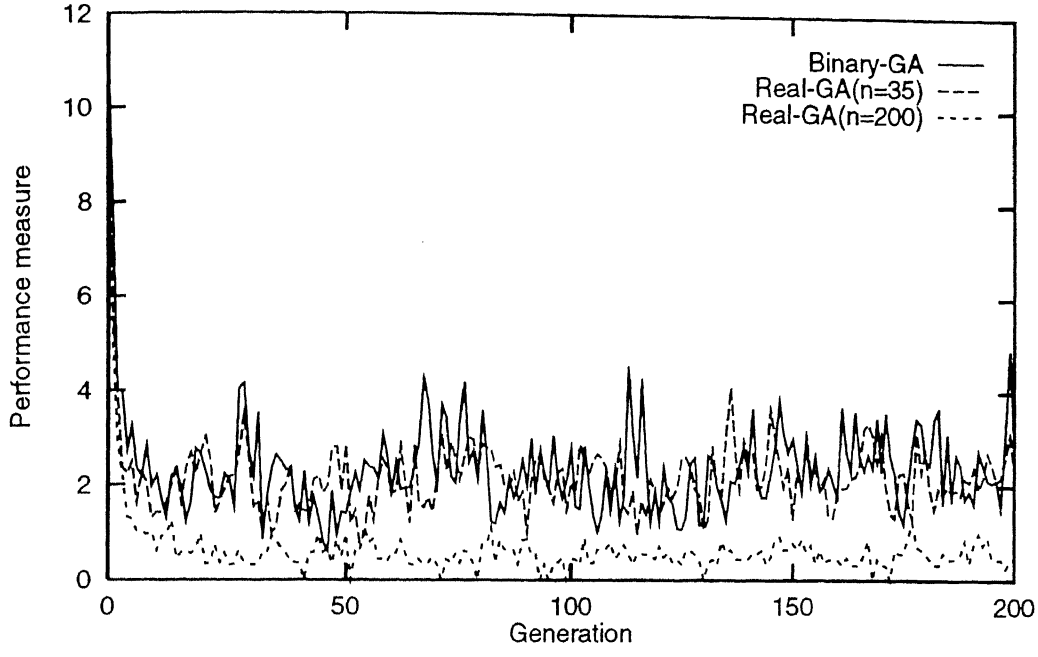


Figure 4.35: Performance measures of binary and real-coded GAs(Function MM4)

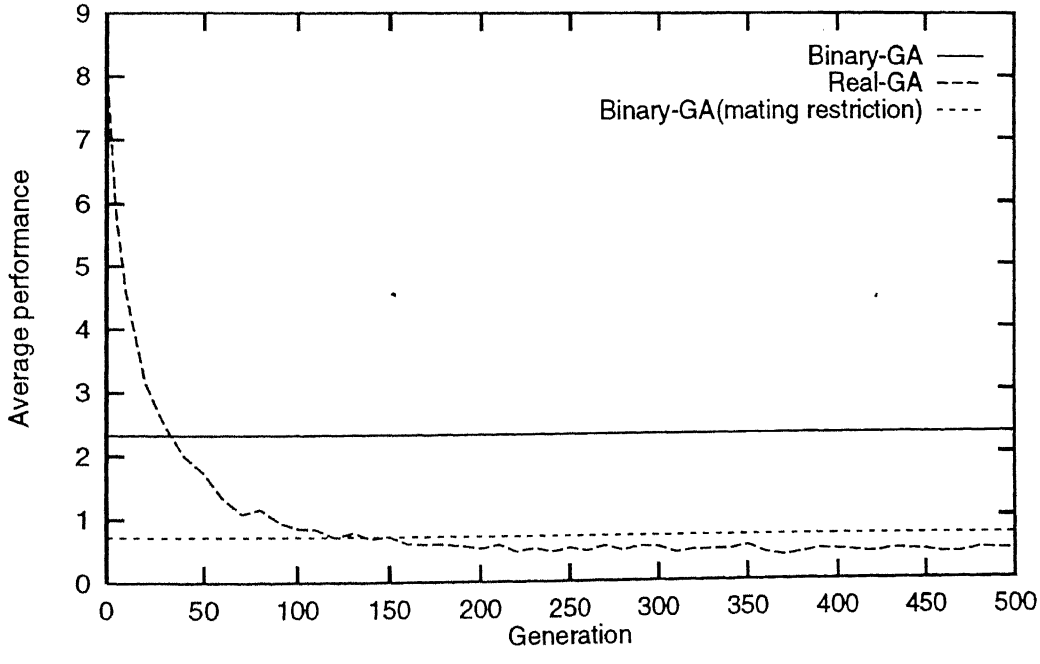


Figure 4.36: Average performance measures of binary and RGA (Function MM4)

All the peaks are expected to receive $\frac{1}{4} \times 100 = 25$ points. The variance of each peak is $\sigma_i^2 = 18.75$ while for non-peak region, it is $\sigma_5^2 = 75.0$. After application of sharing, the population is distributed over the peaks (Figures 4.39 and 4.40). In order to prevent some individuals to fall on non-peak area, the mating restriction scheme has been applied to both GAs in the crossover operation (Figures 4.41 and 4.42). The performance measures of binary-GA and real-GA ($n = 35$) are similar, whereas real-GA ($n = 200$) shows improved performance measure (Figure 4.43). Figure 4.44 reveals that real-GA with $n = 20$ onwards can replace binary-GA with mating restriction scheme.

4.5.6 Function MM6

These random bomodal functions have been created to demonstrate the ability of RGA with SBX and binary-coded GAs. Both the GAs are able to get the optimal results in these functions. For all the fifty functions, the average performance measures from $100th$ to $200th$ generation have been calculated using real-coded and binary-coded GAs. Figure 4.45 shows the plot between the function number and the corresponding average performance measure. The results of RGA with SBX (n=200) are better than that of binary-coded GAs. In RGA, a higher value of n allows the creation of children points in the proximity of the parents. All the RGA results are closer to the ideal distribution, while that of binary-GAs are worse due to sharing without mating restriction. These random function results confirm that RGA can give better performance with higher values of distribution index(n) and there is no need to use the mating restriction scheme separately.

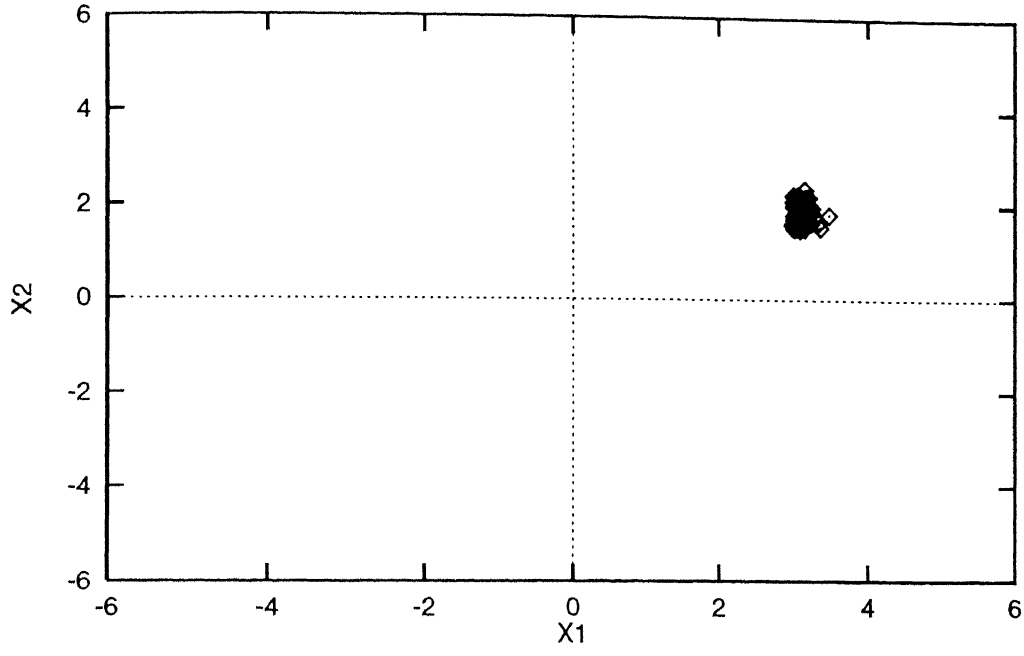


Figure 4.37: Individuals after 200 generations in binary-GA (no sharing) on MM5

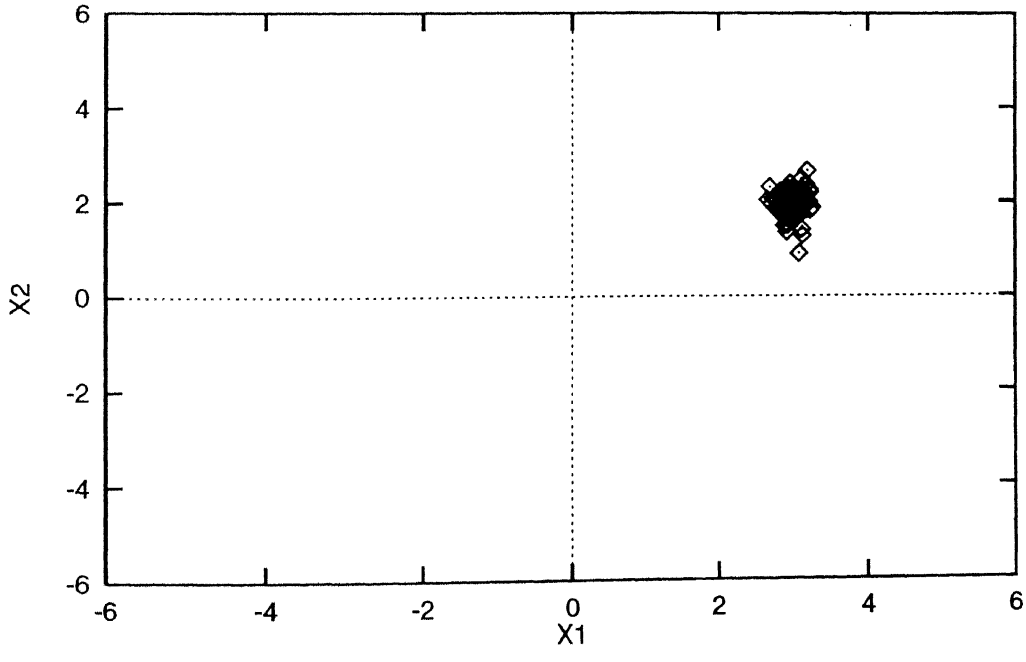


Figure 4.38: Individuals after 200 generations in real-GA (no sharing) on MM5

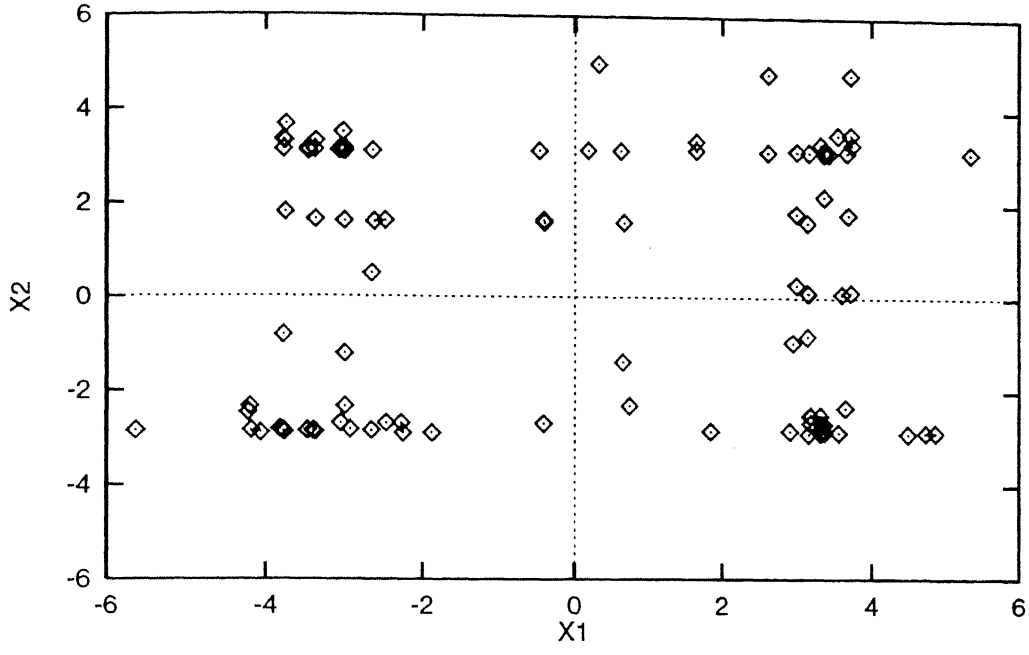


Figure 4.39: Individuals after 200 generations using sharing in binary-GA on MM5

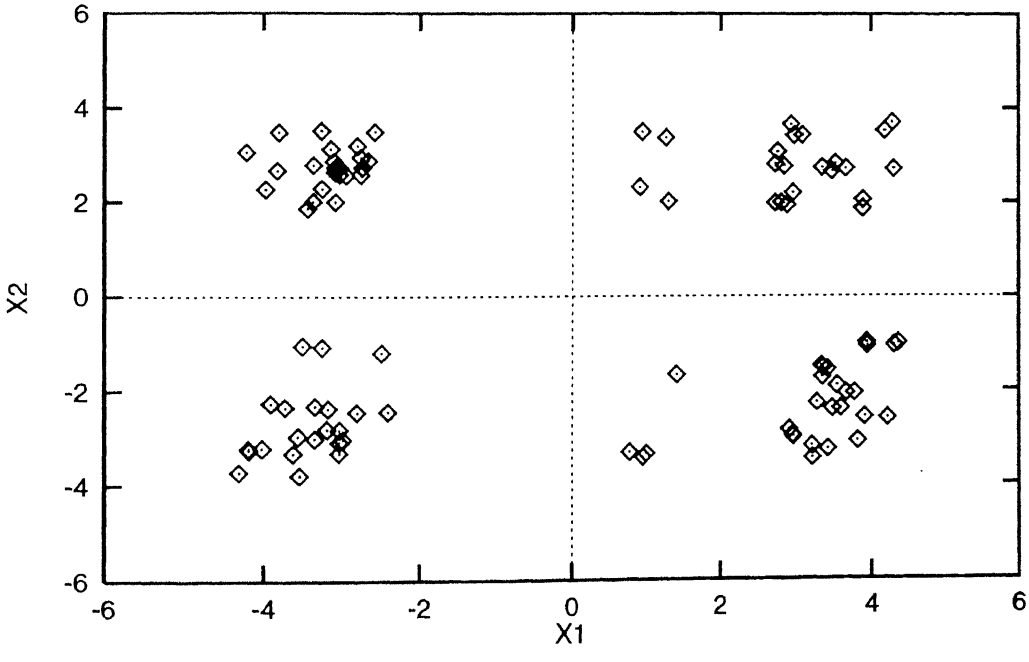


Figure 4.40: Individuals after 200 generations using sharing in RGA (n=35) on MM5

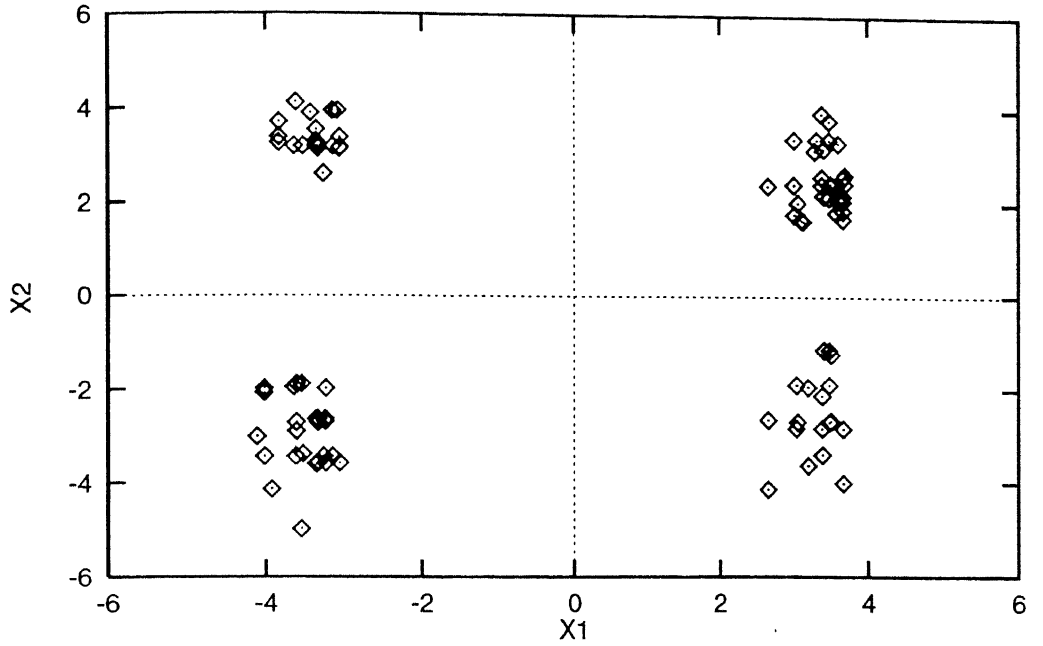


Figure 4.41: Points after 200 generations using mating restriction in binary-GA (MM5)

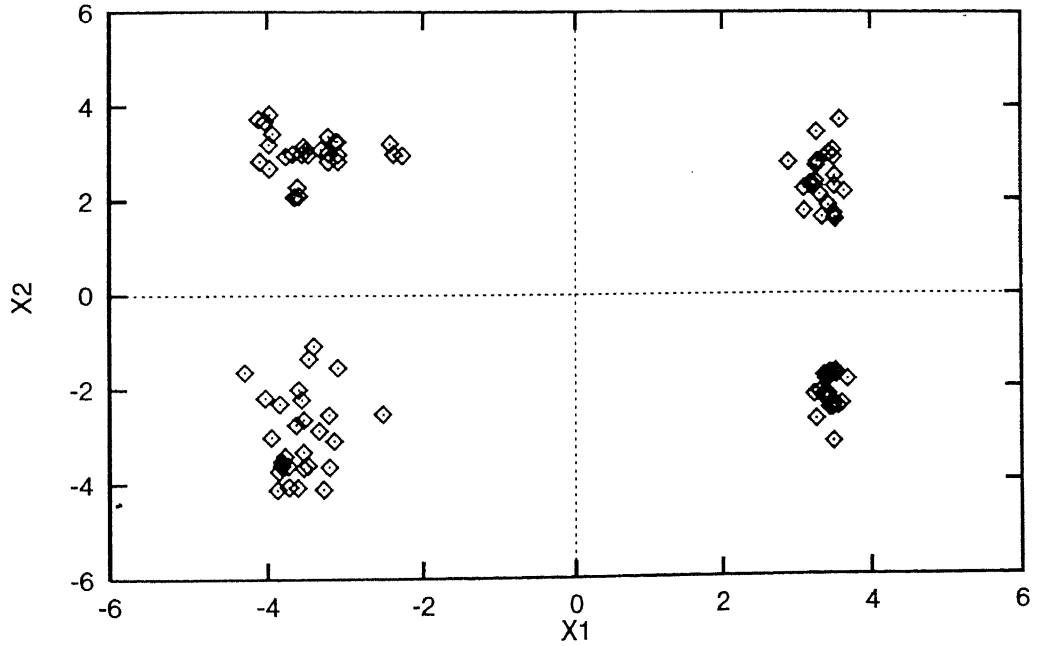


Figure 4.42: Points after 200 generations using mating restriction in RGA (n=35) on MM5

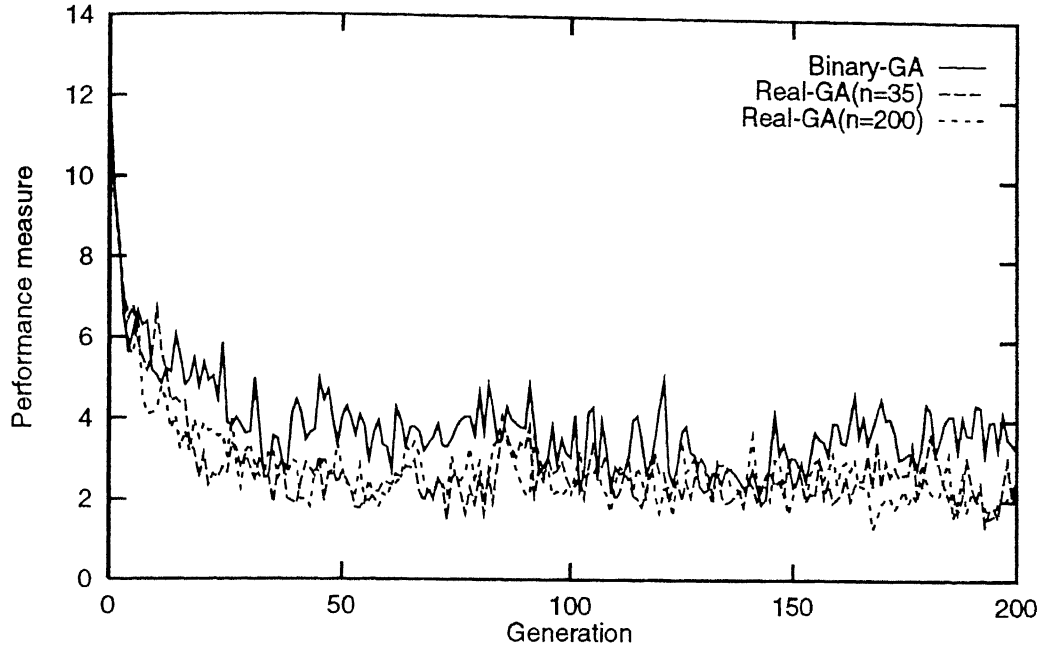


Figure 4.43: Performance measure of binary and real-coded GAs(Function MM5)

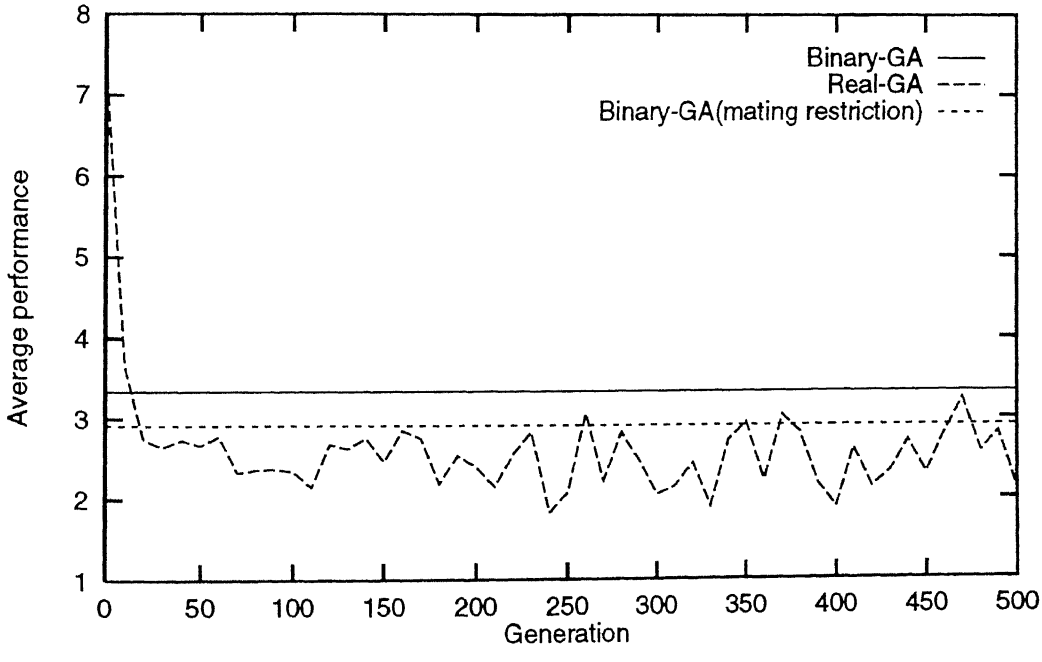


Figure 4.44: Average performance measures of binary and RGA (Function MM5)

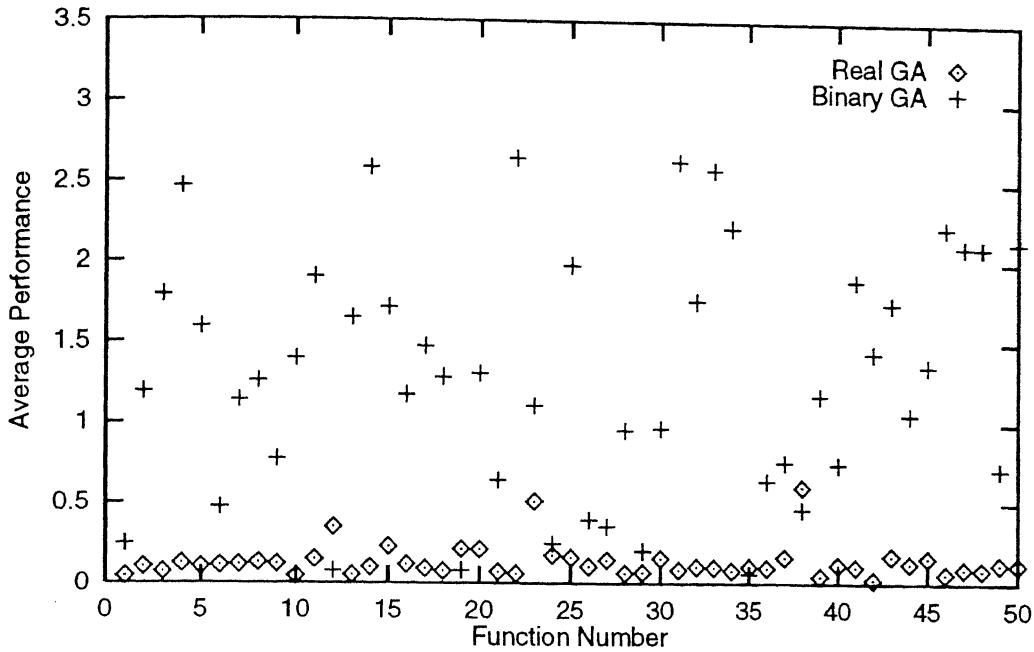


Figure 4.45: Average performance measure of binary and RGA (Function MM6)

4.6 Fractional Sharing

Multimodal optimal solutions are achieved by applying sharing scheme among the individuals. This scheme, however, is accompanied by a criticism that for each individual in the population the sharing function has to be calculated for every other individuals present there (Smith, Forrest, and Perelson, 1992, Goldberg and Richardson, 1987). For population size N this requires N^2 function evaluations. However, if proper bookkeeping is maintained, this can be reduced to half because the sharing function is cumulative ($Sh(d_{ij}) = Sh(d_{ji})$). Goldberg and Richardson (1987) suggested to use smaller subpopulations for calculating the cumulative proximity. In this section, the functions MM1 to MM5 have been tested by sharing performed with varying share size.

For each individual in the population, a small subpopulation P_η of size η (*share size*) is chosen at random and the niche count m_i' is calculated for the individual based

on the random subpopulations. Sharing strategy remains the same as before except Equation 4.2 is replaced by the following equation :

$$m_i' = \sum_{\substack{j=1 \\ j \in P_\eta}}^{share\ size} Sh(d_{ij}). \tag{4.13}$$

If the share size (η) is equal to the total population size (N), the above equation reduces to the equation 4.2.

Both the real-coded and binary-coded GAs have been tested for different share size for the functions MM1 - MM5. The value of the distribution index (n) is taken the same as that in previous sharing results. The performance measures of binary-GAs and real-GA for 1%, 3%, 5%, and 100% share size of the population for function MM1 have been shown in the Figures 4.46 and 4.47, respectively. For 5% share size both the GAs are able to give performance measures similar to 100% share size. So there is no need to calculate the sharing function for all the individuals. For function MM2 the performance results have been taken for 1%, 2%, 20%, and 100% share size (Figures 4.48 and 4.49). In this function 20% share size can replace the 100% share size. For function MM3 1%, 3%, 10%, 100% share size performance results have been shown in the Figures 4.50 and 4.51. Mere 10% share size is sufficient for the sharing function calculations in MM3. Function MM4 (Figures 4.52 and 4.53) also needs 10% share size to give similar 100% sharing results. For function MM5, the sharing results for 5% and 100% random share sizes are equivalent (Figures 4.54 and 4.55).

A comparative analysis of the functions MM1 - MM5 for different share size have been shown in the Figures 4.54 and 4.55. The average performance has been calculated from 400 to 500 generation. Both binary and real GAs show that the critical value of share size is much lower than the full 100% share size (maximum 20% for function

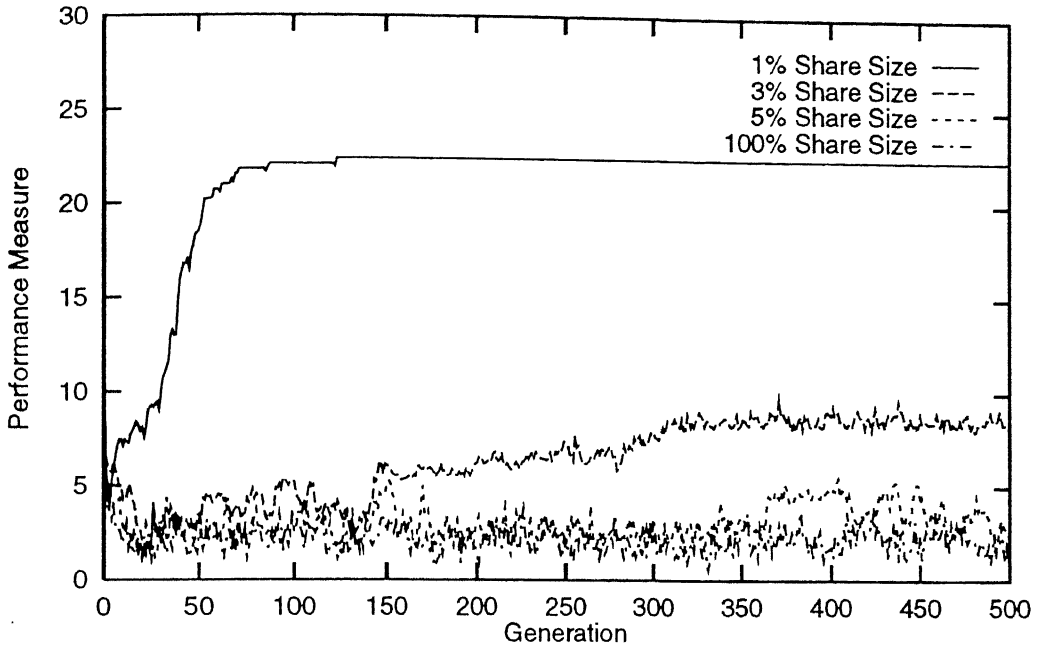


Figure 4.46: Fractional sharing in MM1 using binary-GA

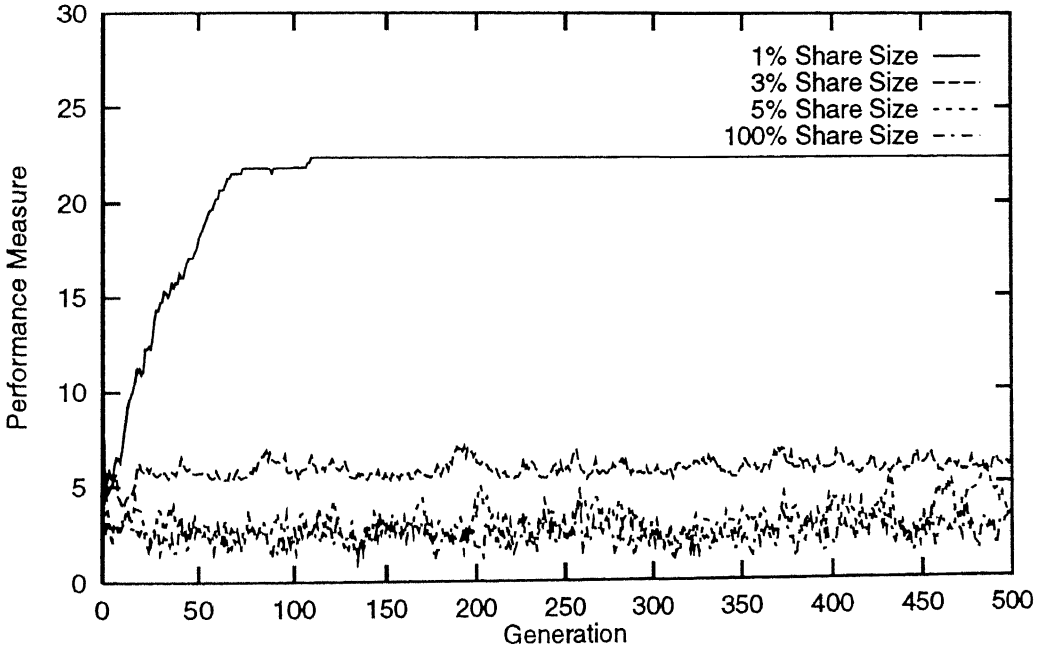


Figure 4.47: Fractional sharing in MM1 using real-GA

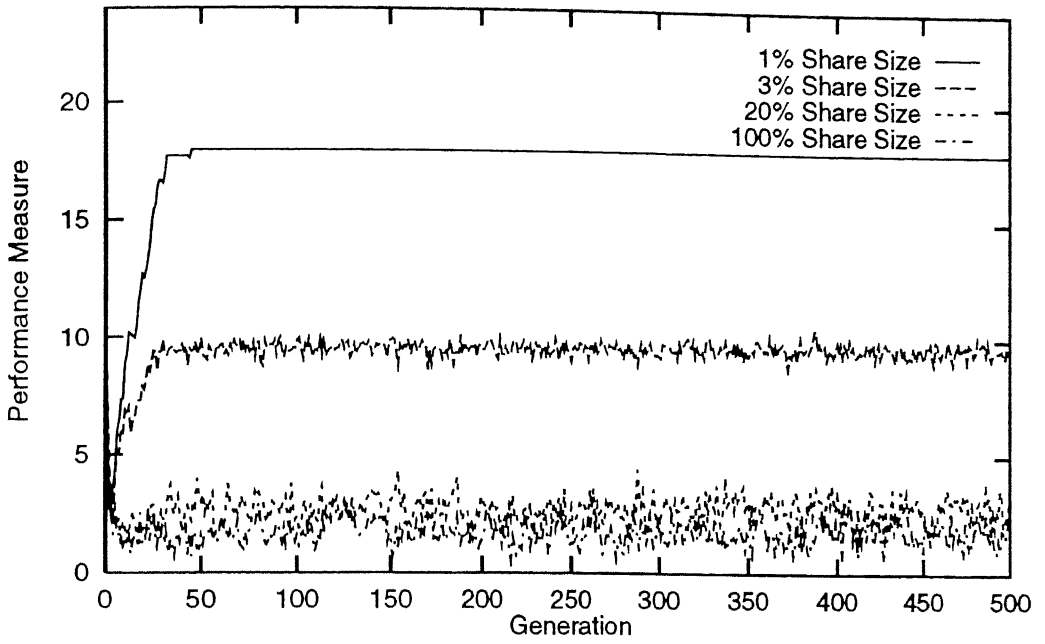


Figure 4.48: Fractional sharing in MM2 using binary-GA

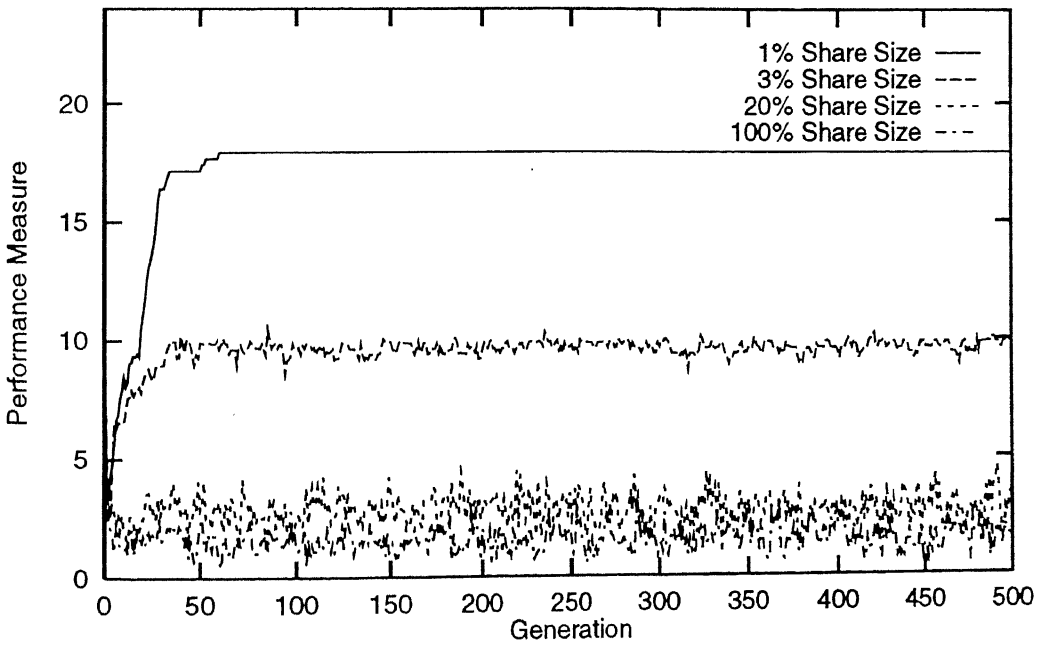


Figure 4.49: Fractional sharing in MM2 using real-GA

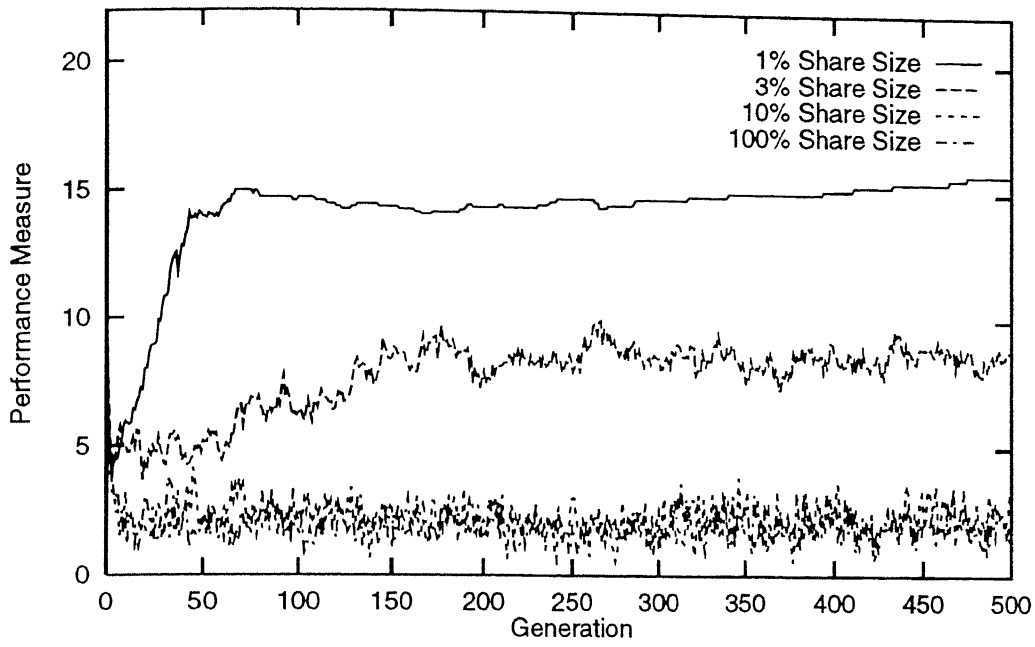


Figure 4.50: Fractional sharing in MM3 using binary-GA

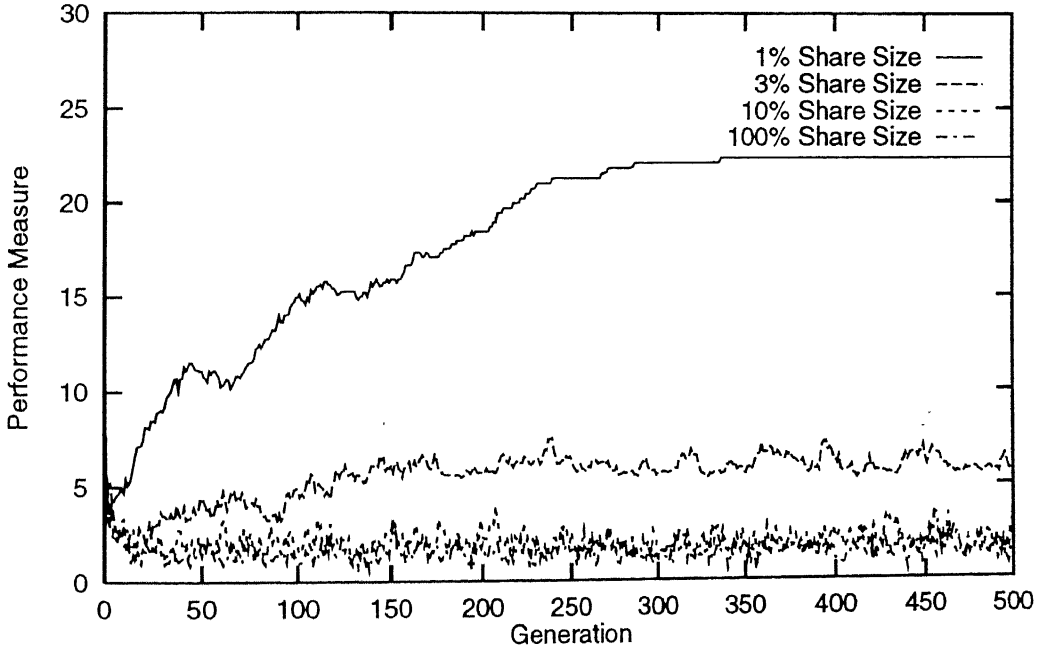


Figure 4.51: Fractional sharing in MM3 using real-GA

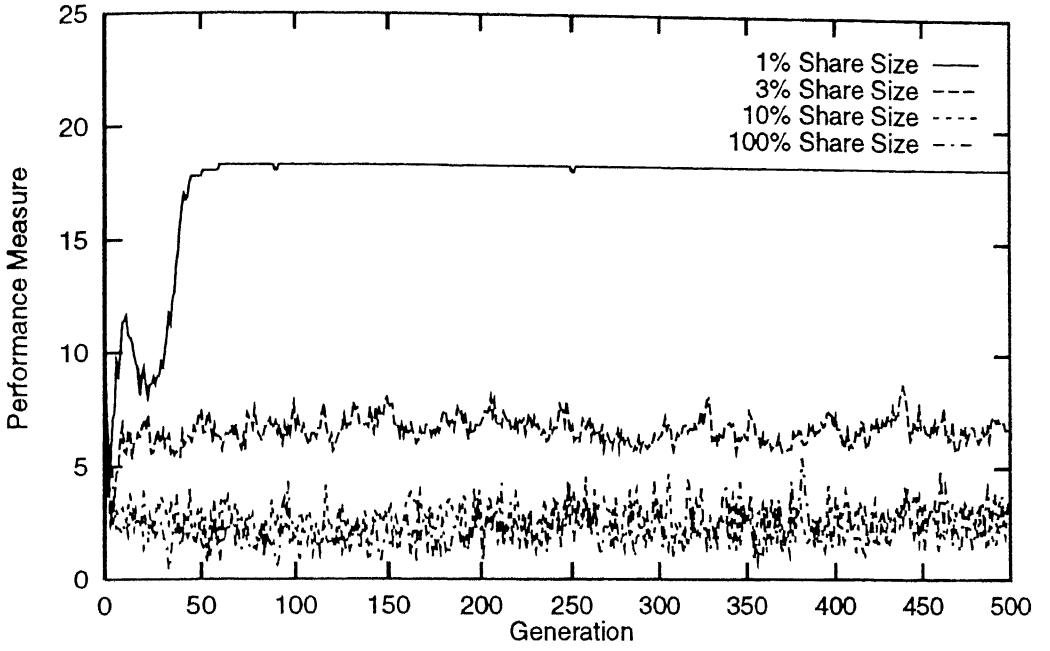


Figure 4.52: Fractional sharing in MM4 using binary-GA

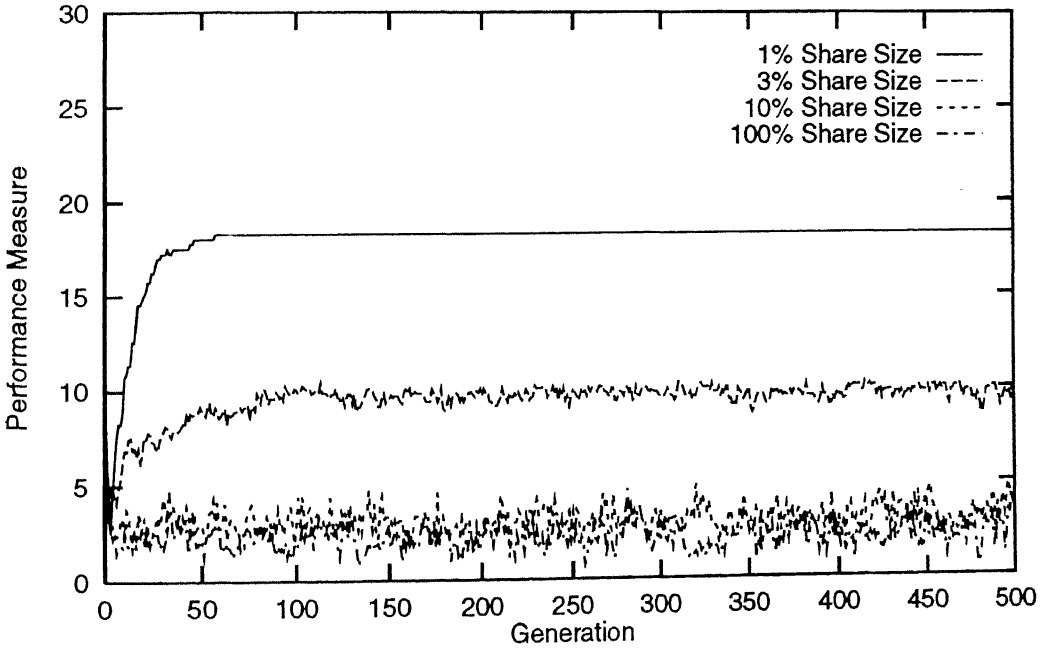


Figure 4.53: Fractional sharing in MM4 using real-GA

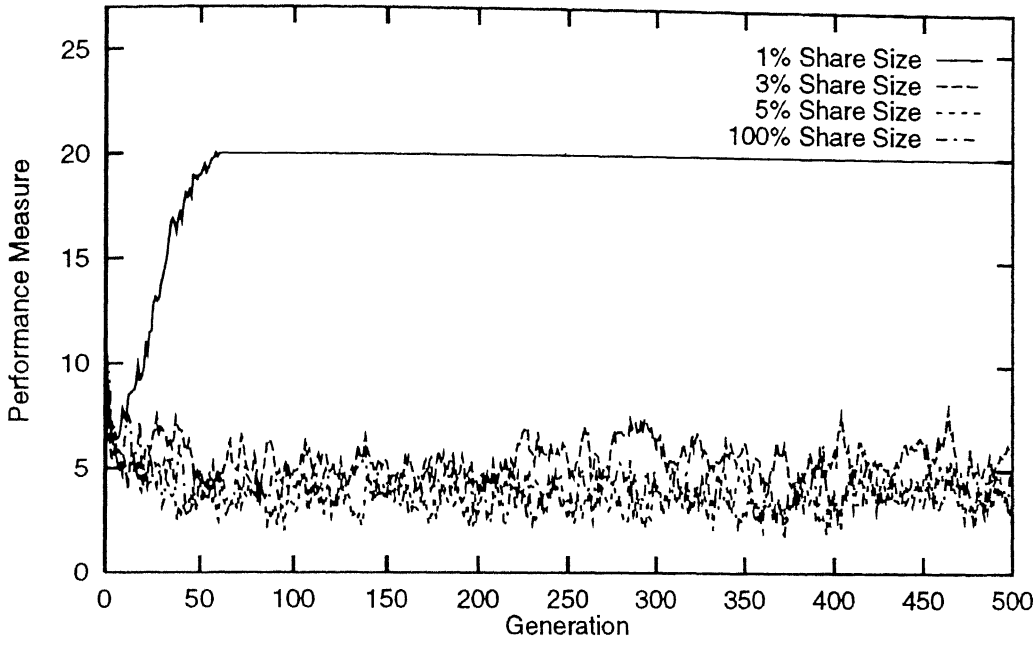


Figure 4.54: Fractional sharing in MM5 using binary-GA

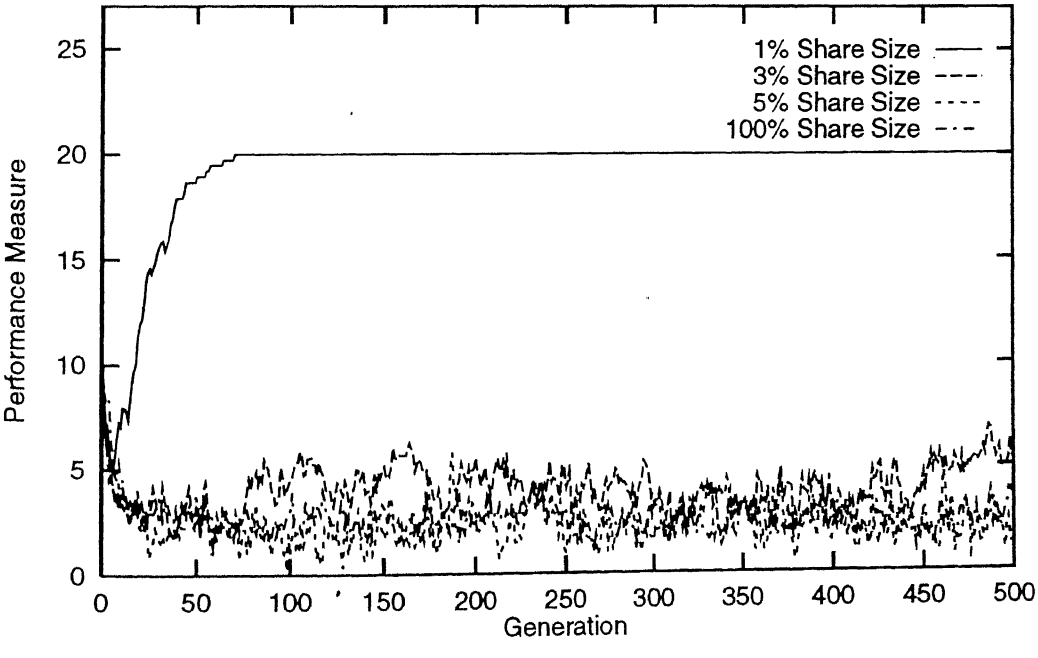


Figure 4.55: Fractional sharing in MM5 using real-GA

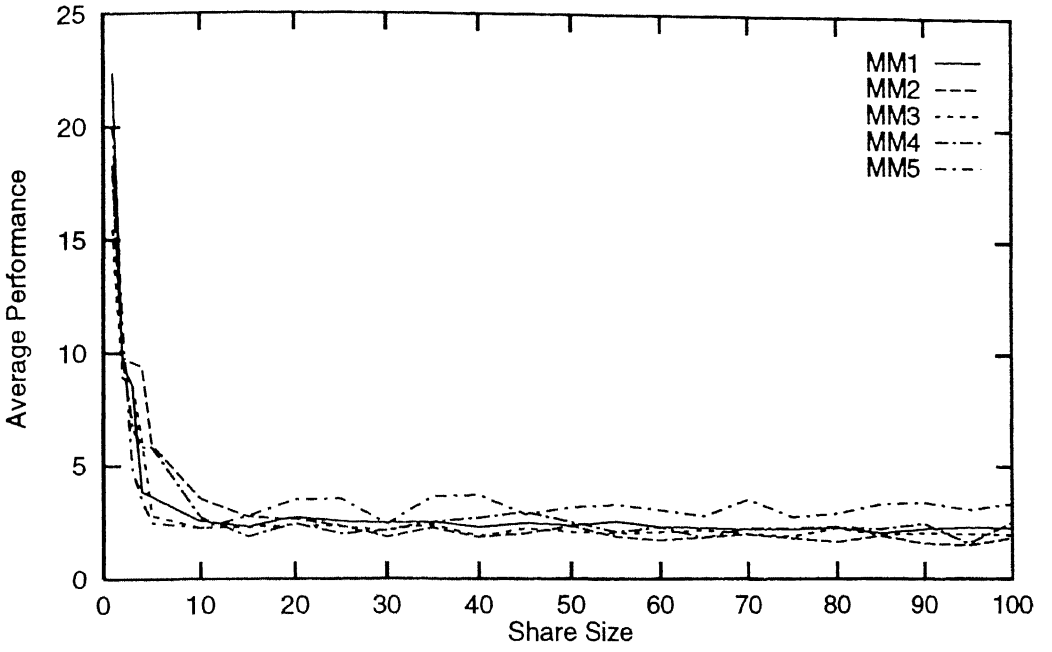


Figure 4.56: Average performances of MM1-MM5 using fractional sharing (binary)

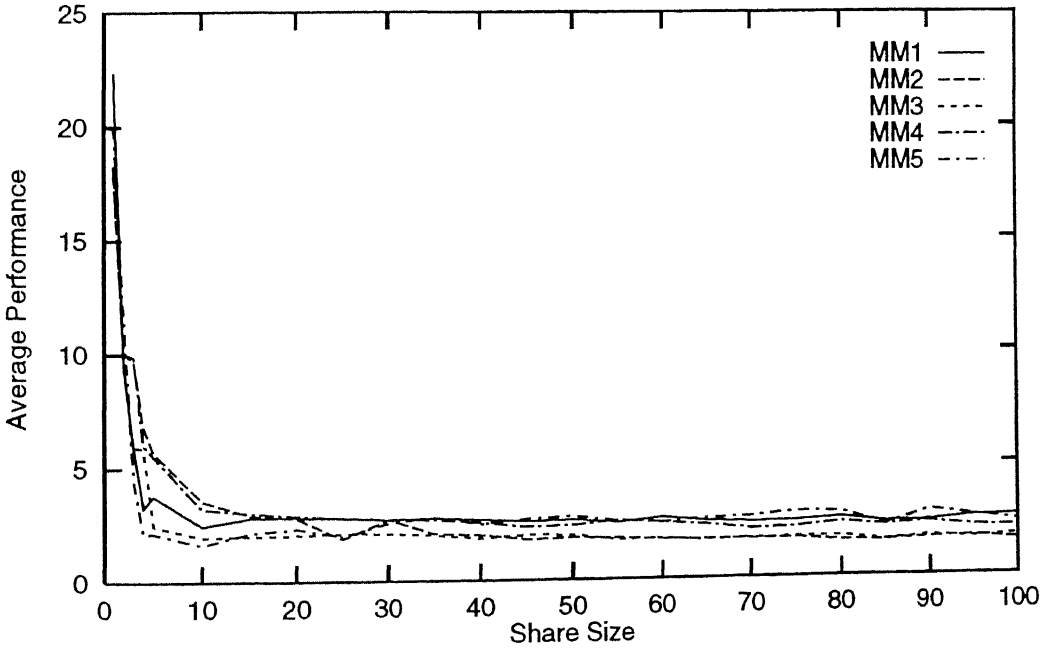


Figure 4.57: Average performances of MM1-MM5 using fractional sharing (real-GA)

MM2). Thus, fractional share size will reduce the N^2 function evaluations to mere an average of 10% or 20% for these type of functions.

4.7 Summary

In order to get multimodal solutions through GAs, sharing should be applied among the individuals in the reproduction operation. Selection of individuals is done on the basis of shared fitness instead of their full potential values. RGA as well as binary GAs are able to explore multimodal solutions in parallel using the sharing principle. With sharing, some points are found on the non-peak area due to wrong selection of mating individuals. The results of sharing can be further improved by applying mating restriction scheme along with it. Mating restriction scheme allows an individual to choose its mating partner on the same peak. If no individual can be selected from the same peak, an individual can pick a random partner from any peak. Based on the peak fitness value, each peak has certain number of expected points to be present on it. The performance measure is calculated on the basis of expected number of points and actual number of points found on all the peaks. RGA with higher values of distribution index (n) is able to give better average performance results than that of binary GAs with mating restriction scheme. This fact has been further supported in 50 random bimodal functions. Sharing concept is critcised for requiring N^2 function evaluations. It has been tried to eliminate this problem by allowing an individual to share its fitness with a fraction of the total population size instead of the whole population. The results show that it is worth checking fractional sharing before going to 100% sharing.

The next chapter presents the implementation of real-coded SBX in multiobjective problems and their comparision with binary results.

Chapter 5

MULTIOBJECTIVE OPTIMIZATION

In a multiobjective optimization problem there are more than one objective functions to be optimized simultaneously. In single-objective optimization problems, the optimum solutions corresponding to the minimum or maximum value of the objective can be found out. But in the multiobjective problem, one global solution may not be achieved in terms of all the objectives at a time. Rather, solutions exist as a set of solutions which may be superior in one or more objectives while not so good in rest of the objectives. All these solutions are accepted solutions of a multiobjective problem and they are known as *Pareto-optimal* solutions or *non-dominated* solutions. Rest solutions are inferior in terms of all objectives and therefore they are known as *dominated* solutions. For a decision maker, it then becomes easier to choose any of the Pareto-optimal solutions for his requirement.

Among the classical methods for solving multiobjective formulation, the *method of objective weighting* comes first. In this method, all the conflicting objectives are

converted into a single objective by considering their relative importance or weightage to the final objective or aim. But the solutions are highly biased on the chosen weightages. Another classical method is *method of distance finctions*. In this method also, the scalarization is done by a demand-level vector usually supplied by a decision-maker. In this case also solution is highly dependent on the demand-level vector. The knowledge of all the individual optimal values is a prerequisite to this method. In *min-max formulation* method the relative deviation of each objective from the individual optimum is minimized. This method can give good results when all objectives of equal importance are to be optimized. This method also requires the knowledge of the problem. The solutions of the classical methods are likely to change if weight factors or demand-levels are slightly changed. For discontinuous search space, these methods may not work suitably. In order to switch over to different solution, these methods are to be applied again and again. But in a real-world problem, decision making can be easily done if a number of alternate solutions are available to the designers or decisison makers.

5.1 Schaffer's VEGA

In 1984, J.D.schaffer tried to implement GA in multiobjective optimization by developing an algorithm known as *Vector Evaluated Genetic Algorithm* (VEGA). He applied the basic recombination operator of simple GA in his algorithm. The entire population was divided into equal subpopulations for all the objectives. Mating pool was created by the best individuals from each subpopulations. In crossover, two mating individuals were selected at random from the mating pool. The motive behind this algorithm was to mate the comptent individuals from different subpopulations but it is more likely

that they may be selected from the same group. VEGA was not quite successful due to its biasness against utopian individuals. The entire population was found to converge on seperate optima, thereby failing to reduce the objective conflicts. In order to avoid the biasness in VEGA, Schaffer introduced two heuristics - the non-dominated selection heuristic and the mate selection heuristic.

In non-dominated selection heuristic he gave importance to nondominated individuals. The dominated points were penalised and the sum of the total penalty was equally distributed among the nondominated individuals as the extra benefit for being good. But this heuristic failed to perform well in case of less number of nondominated individuals. Mate selection heuristic was developed in order to promote the individuals to choose their mating partners from differnt groups. Two individuals were allowed to mate if they were located at maximum Euclidean distance. Mowever this heuristic was not able to stop mating with poor individuals. Later on, he replaced Euclidean distance by *improved distance*. This, too, failed to achieve the desired goal.

5.2 Nondominated Sorting Genetic Algorithm

Goldberg (1989) suggested to minimize the speciation in multiobjective problem by nondominated sorting method followed by sharing among them. Motivated by the above suggestion, Srinivas and Deb (1994) developed an algotrihm called Nondominated Sorting Genetic Algorithm (NSGA).

This algorithm incorporates the same three operators of simple GA but it slightly differs in selection process of individuals. In the selection process all the individuals of the population are compared simultaneously. For minimization process, if all the

objectives of a particular point are greater than that of any point in the population, the first point is then put in the dominated category. This way all the individuals are checked and the non-dominated points are put in the first front. Individuals in the first front are initially given a high dummy fitness value. This dummy fitness assignment is done in order to give equal reproductive ability to all the non-dominated individuals. Then sharing is applied among the first front individuals with reference to their dummy fitness values. So after sharing the dummy fitness value of an individual is reduced depending on the number of points surrounding it in the current front. All the shared dummy fitness values in the current front are compared and the minimum is noted. In further classification of population, the individuals of the first front are ignored and among the rest of the population, non-domiated points are identified and put in the second front. The individuals in the second rank are assigned a dummy fitness which is slightly less than the minimum shared dummy fitness of the previous front points. This is done to promote the individuals from previous good front. Now sharing is done among the members of this front only. This way all the individuals are classified and given a front and shared dummy fitness values.

Reproduction is done on the basis of shared dummy fitness value of an individual. It is evident that the individuals from the better front will receive more number of copies in the mating pool. After forming a mating pool of qualified individuals, crossover and mutation are done as usual. The flow-chart of NSGA is shown in the Figure 5.1. σ_{share} is calculated as described in Chapter4. The only difference in the case of NSGA is that user has to specify the number of peaks q to be induced in Pareto-optimal region. This helps clustering of individuals on different induced peaks. The distribution ability of NSGA is checked by calculating the performance measure as described in Chapter 4.

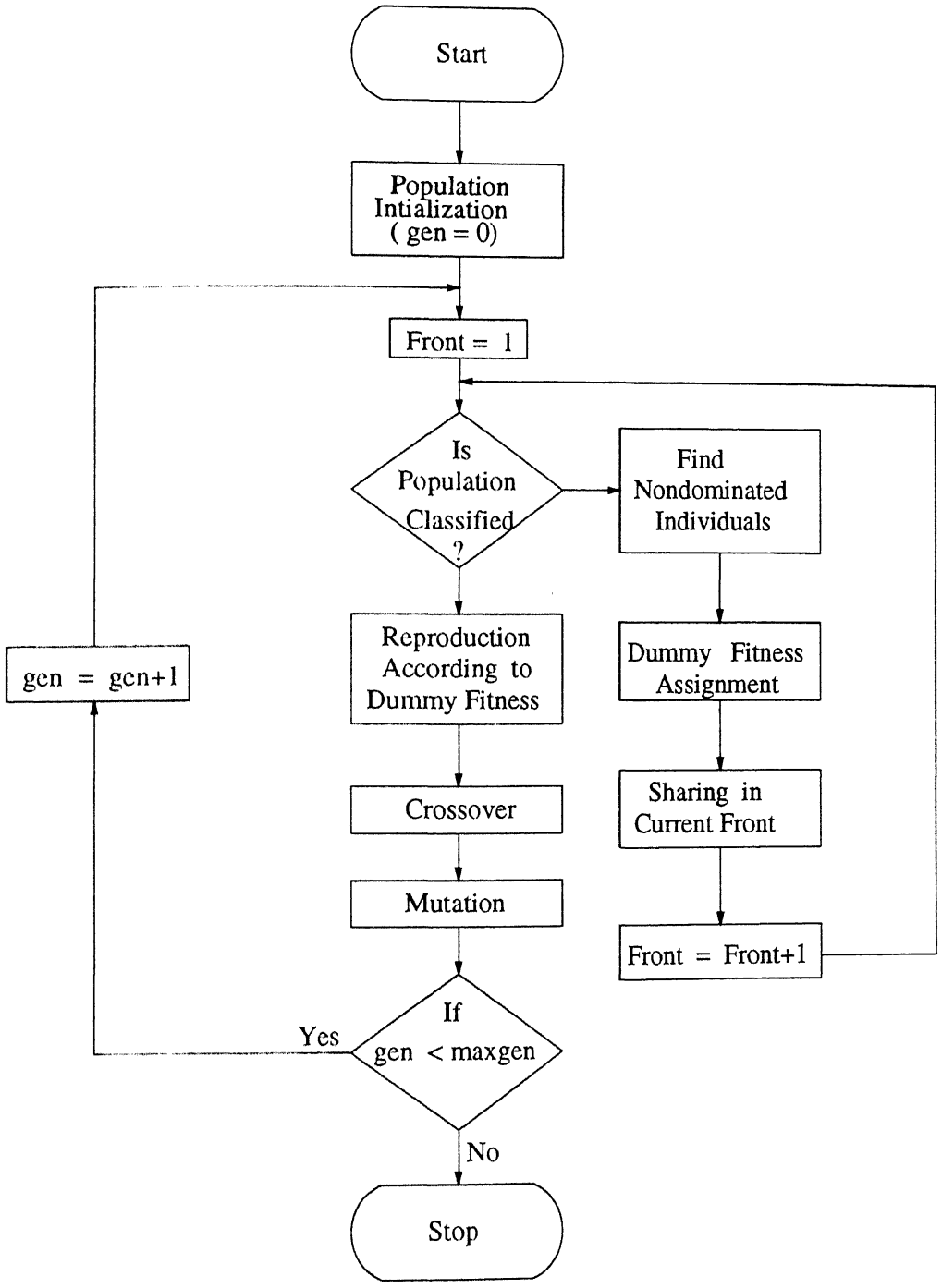


Figure 5.1: NSGA Flow Chart

5.3 Test Functions

Three test functions have been borrowed from Srinivas and Deb (1994) and among these two are single-variable problem with two objectives. The third function is a two variable problem with two objectives. In NSGA, there is no restriction on number of variables and objectives. Through these test functions the distribution of Pareto-otimal regions have been tested using binary-GA and real-coded GA with SBX.

MO1 : Two smooth, single variable, unimodal objectives

$$\text{Minimize} \begin{cases} \text{Fitness1} &= x^2 \\ \text{Fitness2} &= (x-2)^2 \end{cases} \quad \text{Pareto-optimal solution } 0 \leq x \leq 2$$

The two objectives of this function have optimum values at $x = 0$ and $x = 2$ respectively. This is shown in the Figure 5.2 . The Pareto-optimal region is bounded by the two individual optima.

MO2 : A bimodal objective & a smooth unimodal objective

$$\text{Minimize} \begin{cases} \text{Fitness1} & \begin{cases} -x & \text{if } x \leq 1 \\ -2+x & \text{if } 1 < x \leq 3 \\ 4-x & \text{if } 3 < x \leq 4 \\ -4+x & \text{if } x > 4 \end{cases} \\ \text{Fitness2} & (x-5)^2 \end{cases} \quad \text{Pareto solution } \begin{cases} 1 \leq x \leq 2 \\ 4 \leq x \leq 5 \end{cases}$$

The first objective is a C^0-continuous function with two minima at $x = 1$ and $x = 4$. The second objective is a parabola with global optima at $x = 5$. This function is shown in the Figure 5.3 .

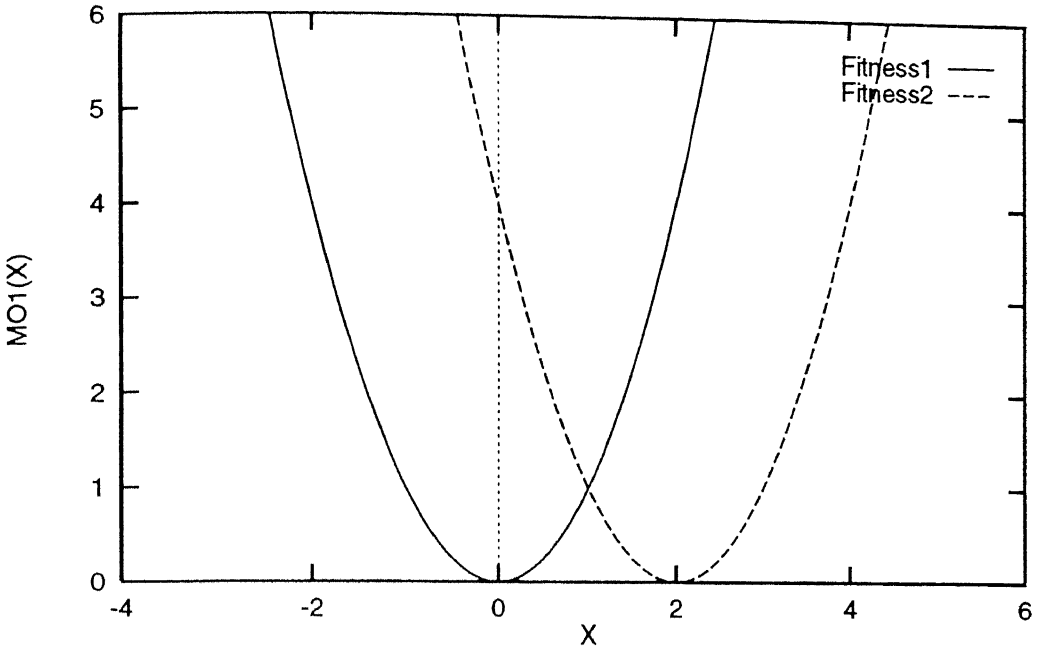


Figure 5.2: Function MO1

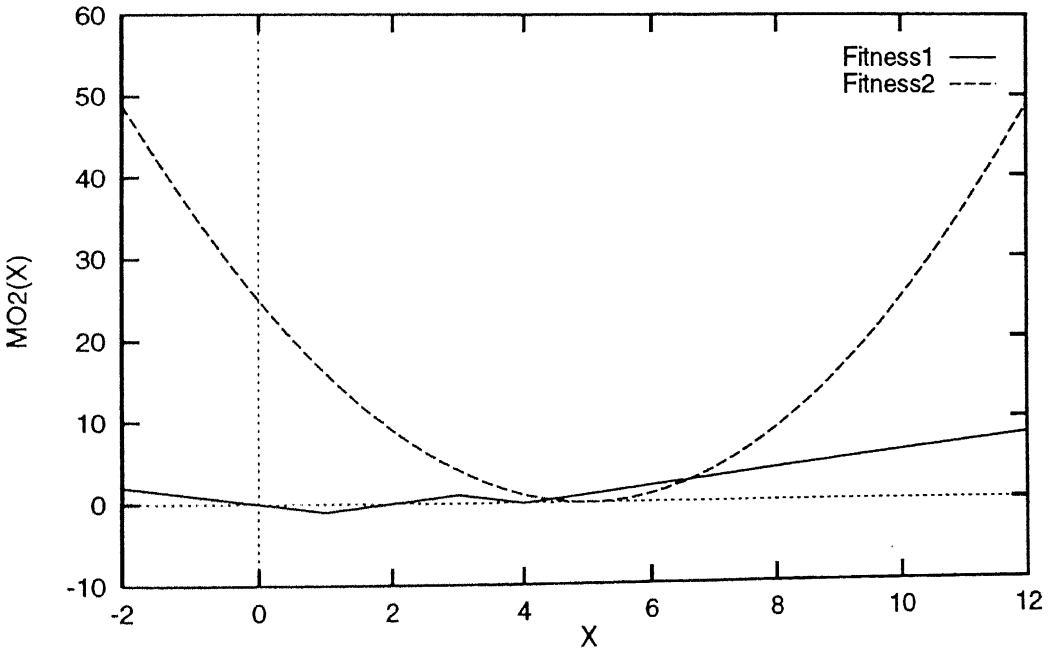


Figure 5.3: Function MO2

MO3 : Two variable, unimodal & monotonically decreasing objectives

$$\text{Minimize} \begin{cases} \text{Fitness1} &= (x_1 - 2)^2 + (x_2 - 1)^2 + 2 \\ \text{Fitness2} &= 9x_1 - (x_2 - 1)^2 \end{cases} \quad \text{Pareto solution at } x_1 = 2.5.$$

The optimal value of first objective is at $x_1 = 2$ and $x_2 = 1$, while for the second objective, it is at $x_1 = 0$ and $x_2 = 1$. This function has been shown in Figure 5.4 . Pareto-optimal points can be found out by equating the slopes of two objectives as :

$$\left(\frac{dx_2}{dx_1}\right)_{\text{fitness1}} = \left(\frac{dx_2}{dx_1}\right)_{\text{fitness2}}$$

$$\text{or,} \quad -\frac{(x_1 - 2)}{(x_2 - 1)} = \frac{9}{2(x_2 - 1)}$$

If we assume that $x_2 \neq 1$ the above expression can be solved to get x_1 = -2.5

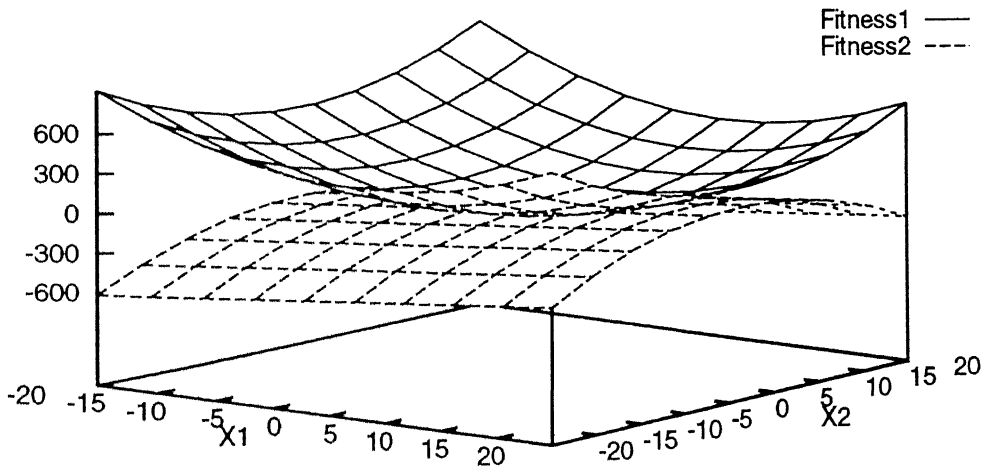


Figure 5.4: Function MO3

5.4 Simulation Results

The following parameters have been taken to compare the results of NSGA using binary-coded and real-coded GAs.

Maximum generation	500
Population size, N	100
String length, l	30
Crossover probability, p_c	1.0
Mutation probability, p_m	0.0
Distribution index, n	30

All the results have been taken in a single run. The stochastic remainder selection method has been used to reduce the stochastic error.

5.4.1 Function MO1

The Pareto-optimal region in this function is bounded by the range $x = 0$ to $x = 2$. The distribution of population in the Pareto-optimal range has been achieved by creating 10 artificial peaks in range (0,2). This range is equally divided into 10 parts for the sake of identifying different peaks. The number of peaks can be changed by the user. So the σ_{share} in this case can be calculated as :

$$\sigma_{\text{share}} = \frac{2-0}{2\times 10} = 0.1.$$

At the begining the entire population are initialized in the range (-10,10). Both GAs are able to get the patero-optimal range and maintain it throughout the generations. The results at 500 generation are shown in the Figures 5.5 and 5.6. However, the nondominated points were found much before the $500th$ generation. Here n=30 has been used for RGA with SBX. The distribution ability of both the GAs have been

checked by calculating their performance measures. All the 10 peaks are expected to get X_i=10 number of expected points. The variance of each peak is $\sigma_i^2 = 9$ while for dominated point it is $\sigma_{11}^2 = 90$. Figure 5.7 shows that both the methods are giving similar performance measures.

5.4.2 Function MO2

This function has two disjoint Pareto-optimal regions. In this function also the intial range of the variable has been set in the range (-10,10). But, both GAs are able to distribute the population in seperate Pareto regions (Figures 5.8 and 5.9), thereby showing the distributing ability of NSGA in disjoint Pareto-optimal regions. Like MO1, we have used $\sigma_{\text{share}} = 0.1$, as 10 artificial peaks are to be induced in the Pareto-optimal range. The range between 1.0 to 2.0 has been equally divided into 5 parts to identify 5 peaks in this range and similar division has been done in 4.0 to 5.0 range to accomodate 5 peaks. The variance of each peak is similar to that in function MO1. The performance measures of binary GA and real GA with n=30 are similar as shown by Figure 5.10.

5.4.3 Function MO3

The Pareto-optimal range in this function has infinite range. But the region has been restricted between $-20 \leq x_1, x_2 \leq 20$. Binary and real coded GAs are able to distribute their population along the line $x_2 = -20$ to 20 at $x_1 = -2.5$. The string length in binary GA has been taken 30, where 15 bits are used to code each variables. The σ_{share} in this case is 8.0. The Pareto results are shown in Figures 5.11 and 5.12. The value of distribution index taken in real-coded GA is 30.

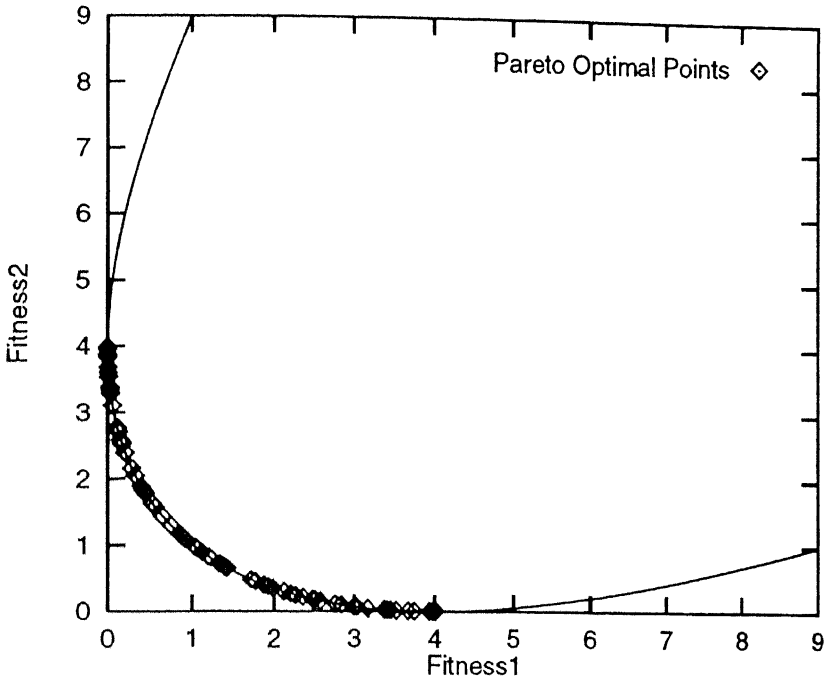


Figure 5.5: Population at generation 500 using binary-GA for MO1

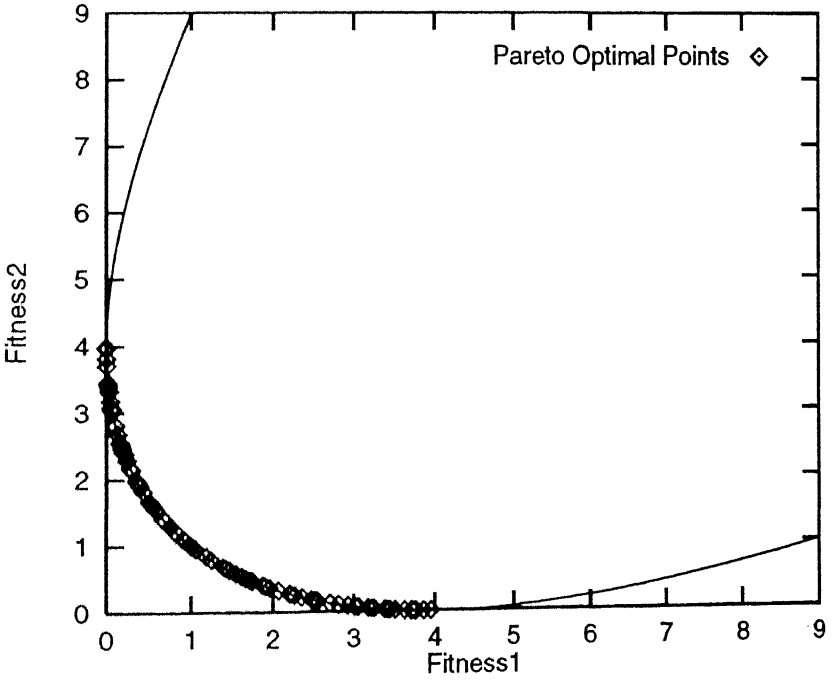


Figure 5.6: Population at generation 500 using real-GA ($n = 30$) for MO1

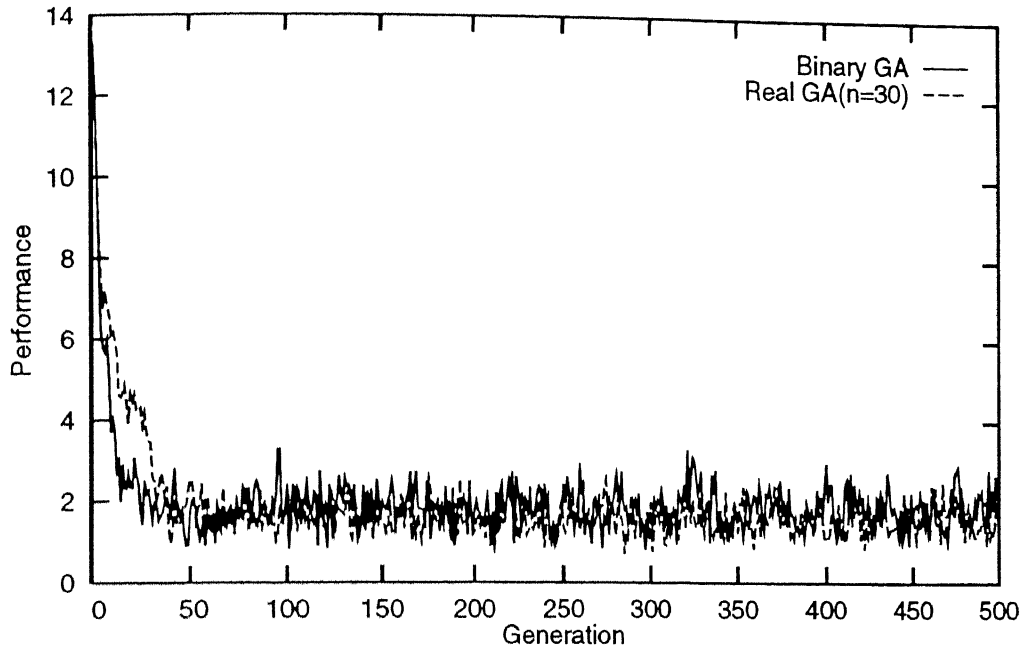


Figure 5.7: Performance Measures of MO1 using binary and Real GA

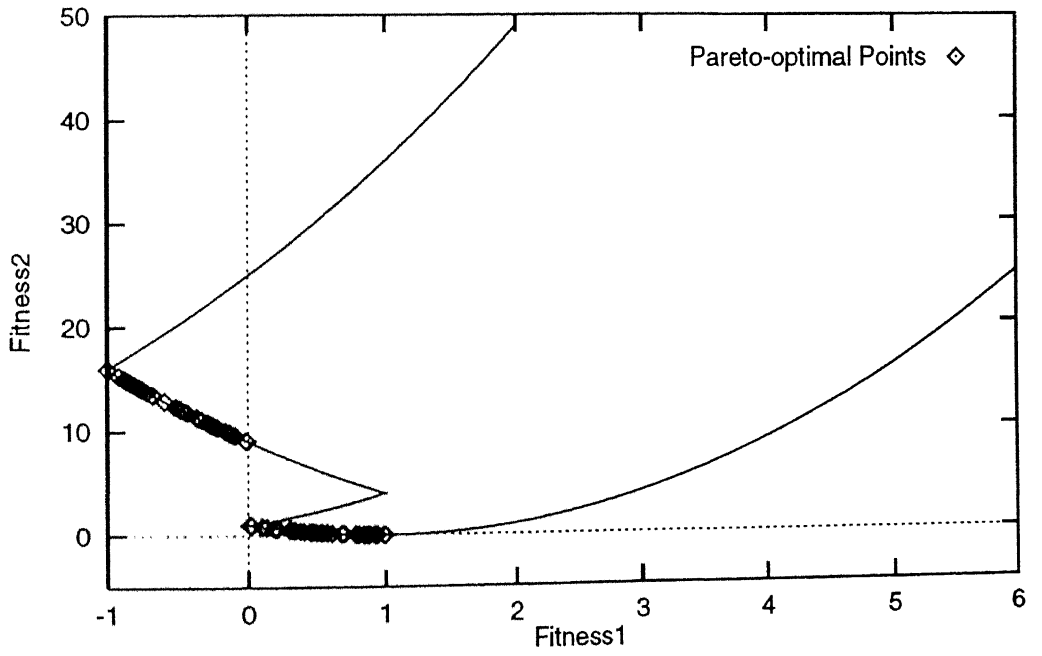


Figure 5.8: Population at generation 500 using binary-GA for MO2

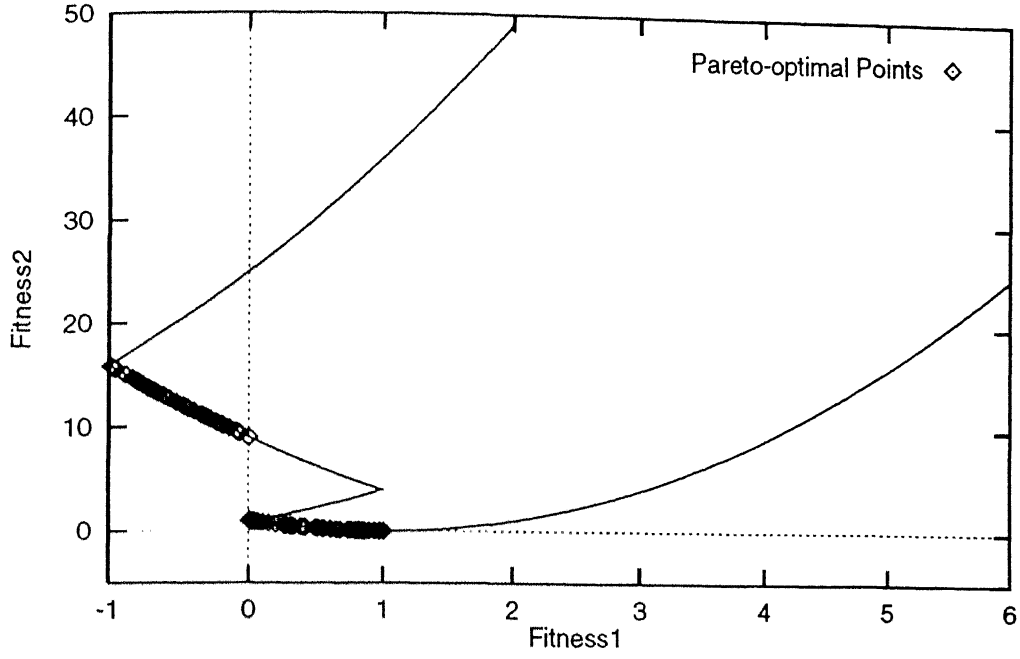


Figure 5.9: Population at generation 500 using real-GA ($n = 30$) for MO2

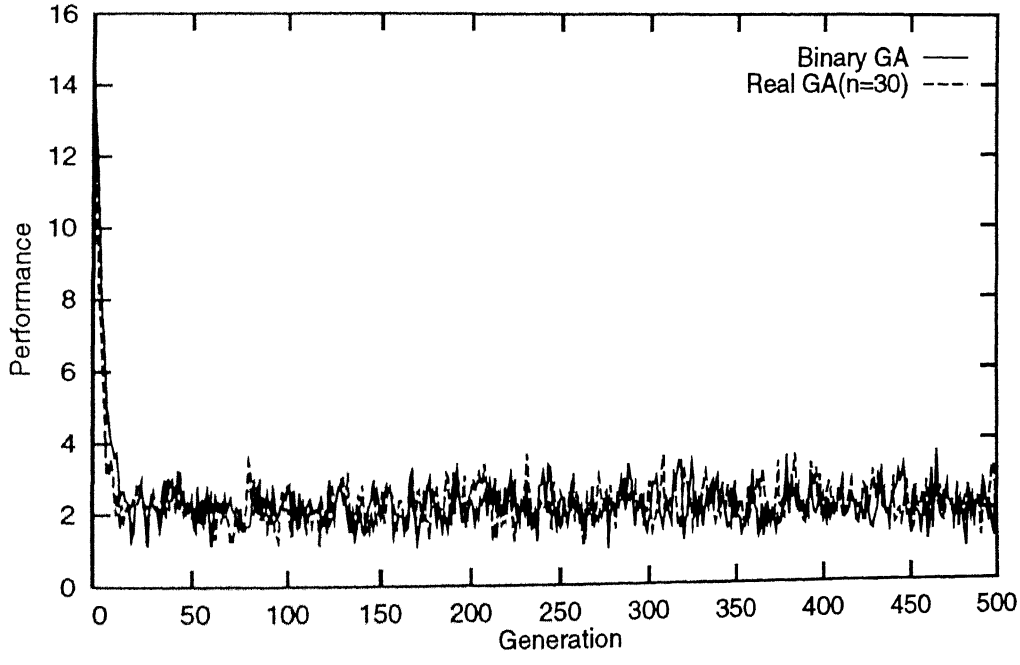


Figure 5.10: Performance measures of MO2 using binary and real GA

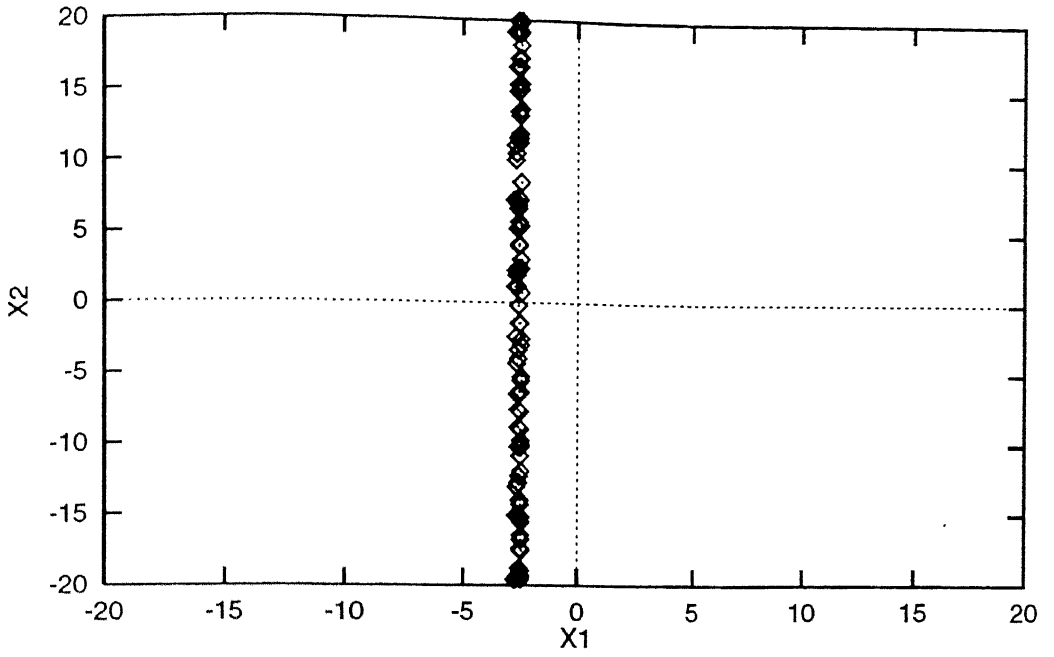


Figure 5.11: Population at generation 500 using binary-GA for MO3

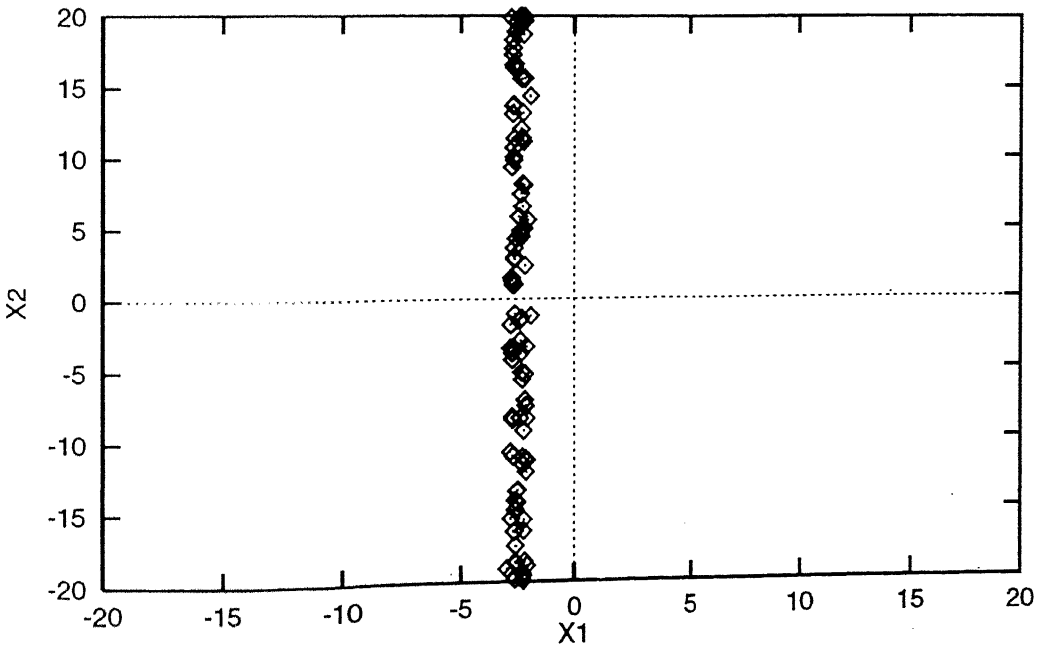


Figure 5.12: Population at generation 500 using real-GA ($n = 30$) for MO3

5.5 Multiobjective Welded Beam Optimization

In order to investigate the efficacy of NSGA in real-world engineering problem, a welded beam design problem has been considered. Binary-coded and Real-coded GAs with SBX have been applied to the problem and a comparision is done.

5.5.1 Problem Formulation

A rectangular cross-sectional beam is welded to a fixed member to carry a given load F (Figure 5.13). In the orginal welded beam problem (Rekalitis, Ravindran, and Ragsdell, 1983), the objective is to minimize the cost of fabrication in terms of a set of variables (h, l, t, and b) subjected to some constraints. These variables are later denoted as x_1, x_2, x_3, and x_4, respectively. The constraint in the original problem regarding the deflection of the free end of the beam is changed to the second objective function. Now the two objectives of the modified problem are conflicting in nature. The minimization of fabrication cost will lead to maximization of deflection and vice versa. The weld beam problem is now given as :

$$\text{Minimize} \begin{cases} \text{Cost} & = & 1.1x_1^2x_2 + 0.048x_3x_4(L+x_2) \\ \text{Deflection} & = & \dfrac{4FL^3}{Et^3b} \end{cases}$$

$$\text{subject to} \begin{cases} g_1(x) & = & S_{yt} - \sigma(x) & \geq & 0, \\ g_2(x) & = & S_{sy} - \tau(x) & \geq & 0, \\ g_3(x) & = & P_c(x) - F & \geq & 0, \\ g_4(x) & = & x_4 - x_1 & \geq & 0. \end{cases}$$

The parameter L is the length of the beam not welded and this is kept fixed to 14 inches. The parameter E is the Young's modulus of the beam. The cost of fabrication includes the material cost and the welding labor cost. The first constraint limits the

bending stress ($\sigma(x)$) anywhere in the welded beam to the maximum allowable strength (S_{yt}) of the beam material. The second limits the shear stress ($\tau(x)$) in the weld to the allowable shear stress (S_{sy}) of the material. The third constraint restricts the buckling of the plate ($P_c(x)$) due to the applied load. The fifth constarint makes sure that the weld size (h) is less than the thickness (b) of the beam. The details of different stresses and loads given by :

$$\text{Bending stress, } \sigma(x) = \frac{6FL}{x_4 x_3^2}.$$

Shear stress is given in terms of primary stress (τ') and secondary stress (τ'') as:

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau'' \cos(\theta) + (\tau'')^2},$$

$$\text{where } \tau' = \frac{F}{\sqrt{2}x_1 x_2}, \quad \tau'' = \frac{MR}{J},$$

$$M = F\left[L + (\frac{x_2}{2})\right],$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_3 + x_1}{2}\right)^2},$$

$$J = 1.414 x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_3 + x_1}{2}\right)^2\right].$$

The parameter M is the moment of F about the center of gravity and J is the polar moment of inertia of the weld group and $\cos(\theta) = x_2/2R$. Buckling load is given by :

$$P_c(x) = \frac{4.013\sqrt{EI\alpha}}{L^2}\left[1 - \frac{x_3}{2L}\sqrt{\frac{EI}{\alpha}}\right],$$

where moment of inertia $I = \frac{1}{12}x_3 x_3^4$, $\alpha = \frac{1}{3}x_3 x_4^3$, $E = 30 \times 10^6$ psi ,and shear modulus $G = 12{\times}10^6$ psi. The allowable shear strength taken is $S_{yt} = 30{,}000$ psi and maximum shear stress is $S_{sy} = 13{,}600$ psi. Load F is 6000 lb.

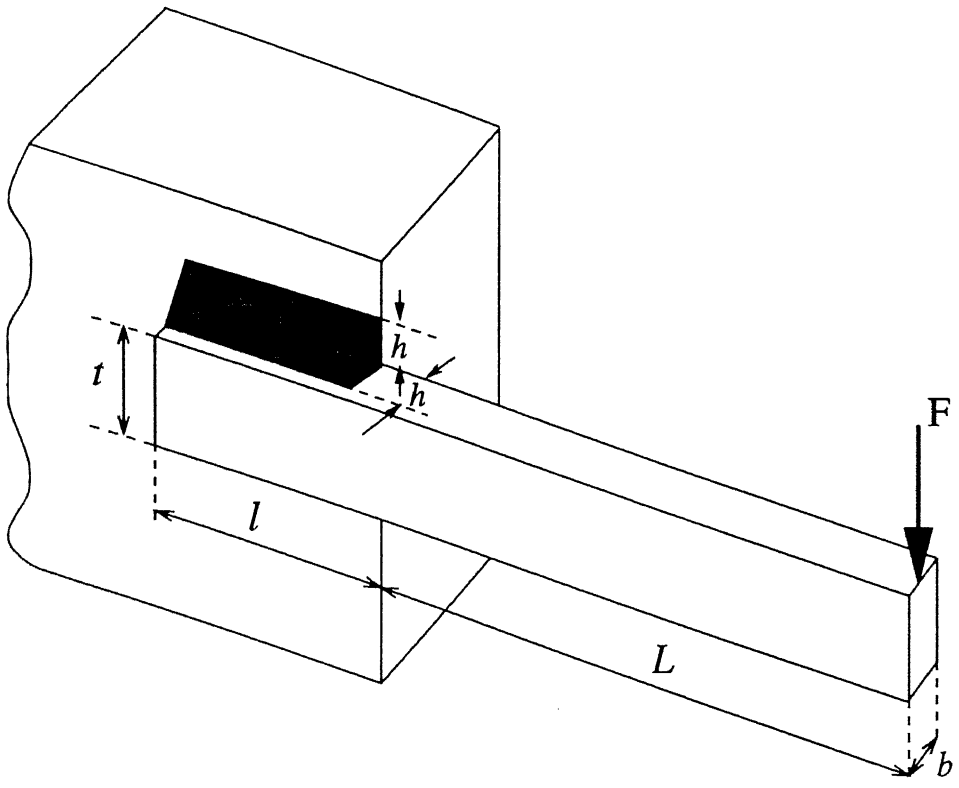


Figure 5.13: Welded Beam

5.5.2 Simulation Results

The following data have been taken for comparing the welded beam problem using binary and real-coded GAs.

Maximum generation	500
Population size, N	100
String length, l	80
Crossover probability, p_c	1.0
Mutation probability, p_m	0.0
Distribution index, n	30

The string length in the binary-coded GA is 80, where 20 bits are used to code each variable. The ranges of the four variables of the beam are given as follows :

$$\begin{aligned} 0.125 &\leq x_1 \leq 5, \\ 0.100 &\leq x_2 \leq 10, \\ 0.100 &\leq x_3 \leq 10, \\ 0.125 &\leq x_4 \leq 5. \end{aligned}$$

In order to create 10 artificial peaks, the sharing distance, σ_{share} can be calculated as:

$$\sigma_{\text{sahre}} = \frac{\sqrt{(5-0.125)^2+(10-0.1)^2+(10-0.1)^2+(5-0.125)^2}}{2\times 10^{1/4}} = 4.38.$$

The Pareto-optimal solution obtained by both the GAs at the end of 500 generation are shown in Figures 5.14 and 5.15 . Although the Pareto-optimal solutions are found within first 50 generation, the simulation is prolonged for 500 generations to investigate whether the GA can maintain stable set of solutions long after they were initially discovered. The extreme Pareto-optimal solutions of the two GAs have been given in Table 5.1. Both the GAs are able to find the low cost solutions but the real-GA with SBX is able to find out the highest cost solution which is approximately 2 times that of the corresponding binary GA solution. The other solutions reveal that the Pareto-optimal solutions vary in the value of variable b only. However, none of the Pareto-optimal solution can be considered as the *absolute* best solution. In some applications, cost could be the main factor, whereas in some cases deflection could be the main factor. In an engineering design, either the cost or the deflection alone may not be the only criteria, a combination of both is sometimes a compromised solution. Finding a number of such optimal solutions simultaneously provides a better insight to the complex interaction of conflicting objectives governing the design. Traditional methods are handicapped in this aspect, since they are expected to find only one solution in one run.

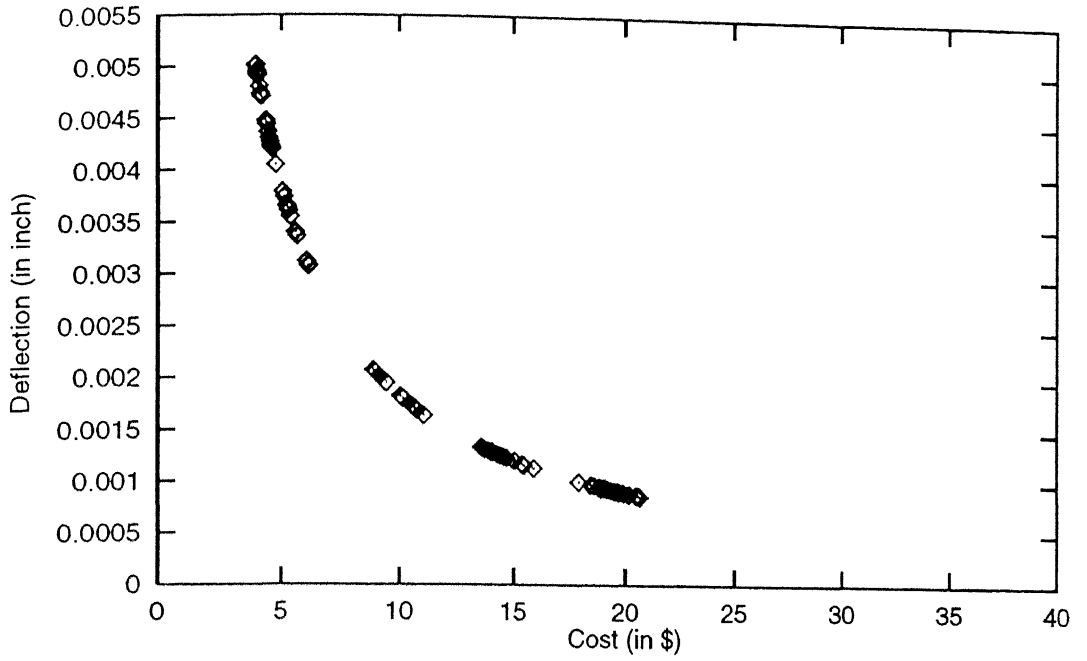


Figure 5.14: Pareto-optimal points of welded beam using binary-GA

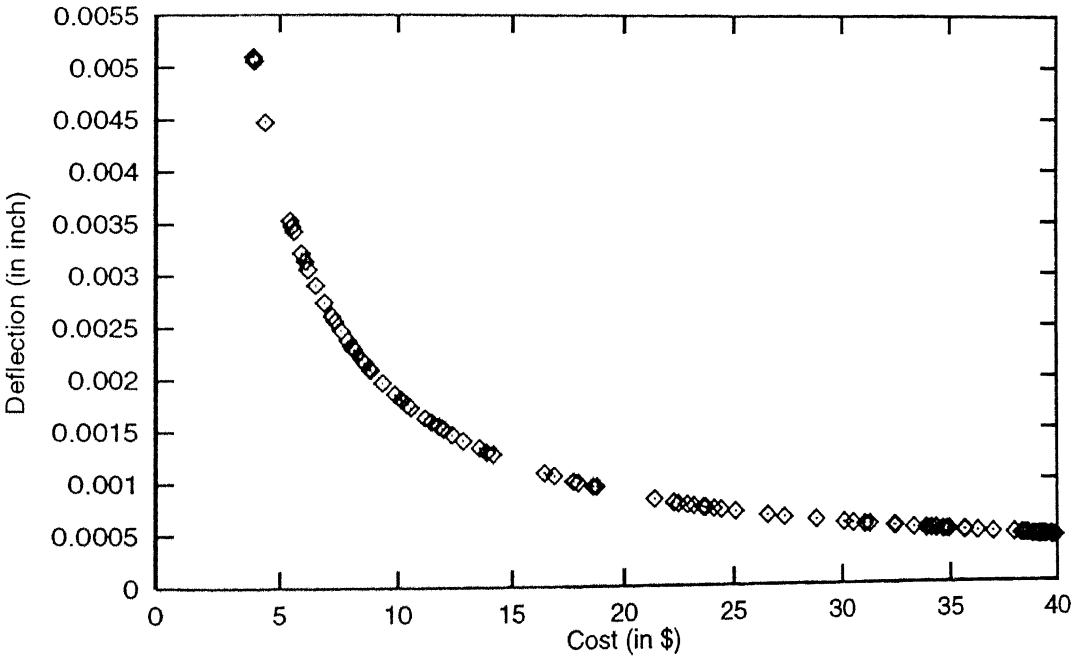


Figure 5.15: Pareto-optimal points of welded beam using real-GA ($n = 30$)

GA	h	l	t	b	Cost	Deflection
Binary	0.4215	2.4649	9.9900	0.4385	3.9440	0.00502
	0.4216	2.4649	9.9899	2.5579	20.6774	0.00086
Real	0.4232	2.4569	9.9824	0.4328	3.9064	0.00510
	0.4276	2.4665	9.9828	4.9871	39.9384	0.00044

Table 5.1: Extreme Pareto-optimal welded beam results of binary and real GA

5.6 Summary

In trying to solve multiobjective problems, classical methods fail to provide Pareto-optimal results as they are highly dependent on weight factors or demand-level vectors chosen. J. D. Schaffer developed an algorithm VEGA by applying tripartite GA, but he could not overcome the biasness against the utopian individuals. He tried to modify the results by two heuristics-mate selection heuristic and nondominated selection heuristic. But these could not help in getting satisfactory results. Srinivas and Deb (1984) developed an algorithim, NSGA, by applying the concept of Goldberg. NSGA differ with simple GA in the reproduction operation only. First, all the individuals are classified according to their nondominated performance in all the objectives and then assigning high equal dummy fitness values to these points. Then sharing is done among the members of current front only. The population is further classified leaving the

previously classified individuals and assigned a dummy fitness value slightly lower than the lowest shared dummy fitness value of the just previous front. This is done till all the individuals are classified. The individuals are given their representation in the mating pool depending on their shared dummy fitness values. Crossover and mutation are done as usual. NSGA has been applied to three test functions and a comparision between binary and real GAs has been done. They gave similar performance. To investigate it further, NSGA has been applied to a popular welded beam problem. In this problem, both GAs are able to find Pareto-optimal solutions, but real-coded GAs with SBX are able to give a broad range solutions than the binary-coded GAs.

Chapter 6

EXTENSIONS

The implementation of real-coded genetic algorithms using SBX for multimodal and multiobjective problems have been done in this thesis. The results have been verified from the existing binary-coded GAs results. The following extension can be done to the present work :

Multimodal problems

- The distribution index, n, has got an important role in the distribution of population in different sub-regions. The distribution ability can be further checked by applying it to more multivariable problems having multimodal solutions.

Multiobjective problems

- The value of σ_{share} is calculated on the basis of number of peaks to be induced in the Pareto-optimal regions, however results can be improved by taking different values of σ_{share}. The addition of mating restriction scheme in sharing may yield good results. σ_{mate} can also be varied to test the results.

- The efficacy of NSGA can be verified by taking more than two objectives where some of them are to be minimized and some of them to be maximized. This can be done by a little change in the classification of individuals.

- Proper selection pressure can be given to NSGA if proper fitness scaling method (Goldberg, 1989) is used so that competent individuals can get more number of their representation in mating pool and the worst one gets zero copy. Tournament selection can also be applied for faster convergence of individuals.

SBX is giving good results in test functions and one engineering design problem, but the robustness of real GAs with SBX can be further checked by applying it to complex engineering problems. In all the cases the role of mutation has been neglected however the results can be tested by taking mutation into consideration. The mathematical support for convergence using SBX can also be pursued.

Chapter 7

CONCLUSIONS

Simualted binary crossover (SBX) operator was developed for continuous search space problems. The SBX operator applied to direct problem variables constituted a search similar to the single-point crossover applied to binary-coded strings representing problem variables. In this thesis, an implementation of the real-coded GAs with SBX operator has been done in finding multimodal and multiobjective problems.

In multimodal problems sharing functions have been used to maintain stable subpopulations around multiple optimum solutions. In a number of test functions, the real-coded GA with sharing has performed similar to the binary-coded GA with sharing. It is advisable to take higher value of n in sharing as it also gives the benefit of mating restriction scheme. By solving 50 different random bimodal problems, it has been observed that the real-coded GA with SBX can better distribute the population members on both optima than the binary-GA. Moreover, the age-old criticism about the full population size requirement of the sharing function method has been disregarded. It has been observed that a small sample of population members (as large as

15% of the population size) can be used to calculate the niche count and still maintain stable subpopulations across the multiple optima.

In the case of multiobjective problems, the concept of nondominated sorting has been implemented in the real-coded GA with SBX operator. Simulation results on three test problems have shown that the real-coded GA can also find and maintain multiple Pareto-optimal solutions in the population. This technique has been further tested to solve an engineering welded beam design problem. Multiple Pareto-optimal solutions corresponding to minimization of both the cost and deflection of the beam are found using the real-coded GA with SBX operatior. The Pareto-optimal range found by real-coded GA in welded beam problem is wider than that obtained by the binary-GA.

References

Agrawal, R. (1995). Simulated binary crossover for real-coded genetic algorithms: Development and application in ambiguous shape modelling, *Master's thesis*, Indian Institute of Technology, Kanpur.

Deb, K. (1991). Optimal design of a welded beam structure via genetic algorithms, *AIAA Journal*, *29*(11), 2013-2015.

Deb, K. (1989). Genetic algorithms in multimodal function optimization, *Master's Thesis*, (TCGA Report No. 89002). Tuscaloosa : University of Albama.

Deb, K. and Agrawal, R. (in press). Simulated binary crossover for continuous search space, *Complex Systems*.

Deb, K. and Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization, *Proceedings of third international conference on genetic algorithms*, Fair-fax, Washington.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*, Reading : Addison-Wesely.

Goldberg, D. E., and Richardson, J. (1987). *Genetic algorithms with sharing for multimodal function optimization*, In J. J. Grefenstette (Ed.), Proceedings of the Second International Conference on Genetic Algorithms (pp. 41-49).

Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M. (1983). *Engineering Optimization - Methods and Applications*, New York: Wiley.

Schaffer, J. D. (1984). *Some experiments in machine learning using vector evaluated genetic algorithms.* Doctoral dissertation, Vanderbit University, Electrical Engineering, Tennessee. (TCGA File No. 00314).

Srinivas, N. (1994). *Multiobjective optimization using nondominated sorting in genetic algorithms.* Master's thesis, Indian Institute of Technology, Kanpur.

Srinivas, N., and Deb, K. (1995). Multiobjective function optimization using nondominated sorting genetic algorithms, *Evolutionary Computation, 2*(3), 221-248.